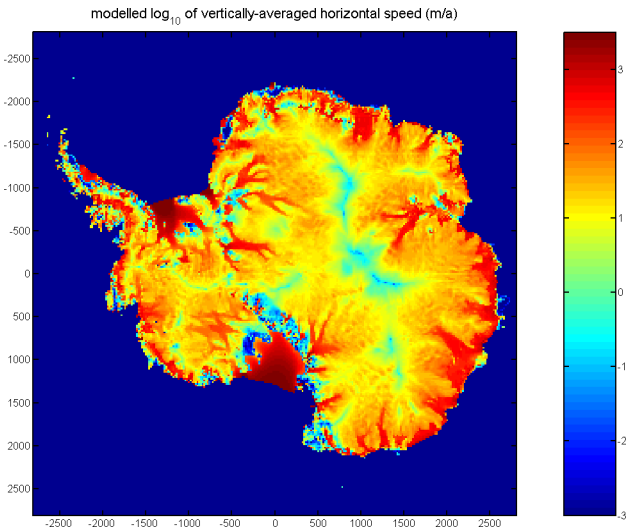
PISM, A PARALLEL ICE SHEET MODEL:

USER'S MANUAL

ED BUELER*, JED BROWN, AND NATHAN SHEMONSKI



Date: October 4, 2007. *ffelb@uaf.edu. Based on PISM revision 191 and PETSC release 2.3.3-p2.
Get PISM by Subversion: `svn co http://svn.gna.org/svn/pism/trunk pism`.

Acknowledgements

The NASA Cryospheric Sciences Program supported this research with grant NAG5-11371. Dave Covey and Don Bahls have been the best possible system administrators for the machines on which we have developed PISM. Thanks to Martin Truffer, Kent Overstreet, and Art Mahoney for helpful comments on the manual and the installation process. We also want to thank several PISM users whose questions and comments have contributed to improving PISM and this manual.

Contents

1. Introduction

Welcome to PISM!

This *User's Manual* describes how to download the PISM source code and install PISM. It describes how to run it for certain simplified geometry situations and certain verification situations. It describes how to use PISM as a modest Greenland ice sheet model or a modest Ross ice shelf model.

But that is all. Users who want to advance the science of ice sheets will need to go beyond what is described here. For such users there are two additional documents to know about:

(1) The *PISM Reference Manual* (`www.dms.uaf.edu/~bueler/refman.pdf`) describes the most important pieces of the source code. It contains, or at least it is intended to contain, the minimum documentation of the PISM source code parts in order to include all the continuum models and numerical methods of PISM.

(2) The *PISM Source Code Browser* within the PISM source code distribution gives a complete view of the class/object structure of the source code. To view it, change directories to `pism/doc/` and do `make`. Then open `doxy/html/index.html` with a web browser.

The *Reference Manual* and the *Source Code Browser* were automatically generated by `doxygen` (`www.doxygen.org`) from comments in the PISM source code. Thus they are not as user-friendly as this *User's Manual*.

WARNING: PISM is an ongoing project. Ice sheet modelling is complicated and not mature (in 2007, anyway). Please don't trust the results of PISM (or any other ice sheet model) without a fair amount of exploration. Also, please don't expect all your questions to be answered here. But do write to us with questions: `ffelb@uaf.edu`.

2. Installation

Installing prerequisities.

1. You will need a UNIX system with internet access. A GNU/Linux environment will be easiest but other UNIX versions have been used successfully. Package management systems are useful for installing many of the tools below, *but* neither PISM itself nor up-to-date PETSc distributions are currently available in the Debian repositories. You will need Python and Subversion installed, but these are included in all current Linux distributions. To use the (recommended) graphical output of PISM you will need an (X Windows server.
2. As PISM is currently under rapid development, we distribute it only as compilable source code. On the other hand, there are several software libraries needed by PISM. Therefore the "header files" for these libraries are required for building PISM. In particular this means that the "developer's versions" of the libraries are needed if the libraries are downloaded from package repositories like Debian. (In summary, `xorg-dev`, `netcdfg-dev`, `libgsl0-dev`, and `fftw3-dev` are the Debian packages we—the PISM developers—have used in compiling and linking PISM.)
3. PISM uses NetCDF (= *network Common Data Form*) for an input and output file format. If it is not already present, install it using the instructions at the webpage or using a package management system.
4. PISM uses the GSL (= *GNU Scientific Library*) for certain numerical calculations and special functions. If it is not already present, install it using the instructions at the webpage or using a package management system.
5. PISM optionally uses the "FFTW" library (FFTW = *Fastest Fourier Transform in the West*) in approximating the deformation of the solid earth under ice loads [11]. If you want the functionality of this earth model, which is coupled to the ice flow and which we recommend, install FFTW or check that it is installed already. If FFTW is *not installed*, however, turn off PISM's attempt to build with it by setting the environment variable `WITH_FFTW=0`. If this library is absent, all of PISM will work *except* for the bed deformation model described in the paper [11].
6. You will need a version of MPI (= *Message Passing Interface*). Your system may have an existing MPI installation, in which case the path to the MPI directory will be used when installing PETSc below. Otherwise we recommend that you allow PETSc to download MPICH2 as part of the PETSc configure process (next). In either case, once MPI is installed, you will want to add the MPI `bin` directory to your path so that you can invoke MPI using the `mpiexec` or `mpirun` command. For example, you can add it with the statement

   `export PATH=/home/user/mympi/bin:$PATH` (for `bash` shell)

   or

   `setenv PATH /home/user/mympi/bin:$PATH` (for `csh` or `tcsh` shell).

   Such a statement can, of course, appear in your `.bashrc`, `.profile`, or `.cshrc` file so that there is no need to retype it each time you use MPI.

From now on this manual will assume use of bash.

7. PISM uses PETSc (= *Portable Extensible Toolkit for Scientific computation*). As mentions of this library will occur frequently in this manual, note "PETSc" is pronounced "pet-see". Download the PETSc source by grabbing the current gzipped tarball at

`www-unix.mcs.anl.gov/petsc/petsc-as/download/index.html`

PISM requires a version of PETSc which is `2.3.3-p2` or later.[1] The "lite" form of PETSc is fine if you are willing to depend on an internet connection for accessing the PETSc documentation.

You should configure and build PETSc *essentially* as described on the PETSc installation page, but it might be best to read the following comments on the PETSc configure and build process first:

(i) Untar in your preferred location, but note PETSc should *not* be configured (next) using root privileges. Note that you will need to define the environment variable `PETSC_DIR` before configuring PETSc (next). For instance, once you have entered the PETSc directory just untarred, `export PETSC_DIR=`\`pwd\`. (Note the use of the backprime (*accent-grave*) character, and not the single apostrophe `'`.)

(ii) When you run the configure script in the PETSc directory, the following options are recommended; note PISM uses shared libraries by default:

```
$  ./config/configure.py --with-shared --with-c-support --with-clanguage=cxx
```

Note that there is no PISM use of Fortran, and that it is sometimes convenient to have PETSc grab a local copy of BLAS and LAPACK rather than using the system-wide version. So one may add "`--with-fortran=0 --download-c-blas-lapack=1`" to the other configure options.

(iii) If there is an existing MPI installation, for example at `/home/user/mympi/`, one can point PETSc to it by adding the option `--with-mpi-dir=/home/user/mympi/`. The path used in this option must have MPI executables `mpicxx` and `mpicc`, and either `mpiexec` or `mpirun`, in subdirectory `bin/` and MPI library files in subdirectory `lib/`.

(iv) On the other hand, it seems common that one needs to tell PETSc to download MPI into a place it understands, even if there is an existing MPI. If you get messages suggesting that PETSc cannot configure using your existing MPI, you might try `configure.py` with option `--download-mpich=1`.

(v) Configuration of PETSc for a batch system requires special procedures described at the PETSc documentation site. One starts with a configure option `--with-batch=1`. See the "Installing on machine requiring cross compiler or a job scheduler" section of the PETSc installation page.

(vi) Configuring PETSc takes many minutes even when everything goes smoothly. A value for the environment variable `PETSC_ARCH` will be reported at the end of the configure process; take note of this value. (Note that a previously installed PETSc can be reconfigured with a new `PETSC_ARCH` if necessary.)

[1] A Debian package for PETSc may exist, but it should only be used if it is version `2.3.3-p2` or later.

(vii) After `configure.py` finishes, you will need to `make all test` in the PETSc directory and watch the result. If the X Windows system is functional some example viewers will appear; as noted you will need the X header files for this to work.

(viii) Finally, you will want to set the `PETSC_DIR` and the `PETSC_ARCH` environment variables in your `.profile` or `.bashrc` file. Also remember to add the MPI `bin` directory to your `PATH`. For instance, if you used the option `--download-mpich=1` in the PETSc configure, the MPI `bin` directory will have a path like `$PETSC_DIR/externalpackages/mpich2-1.0.4p1/$PETSC_ARCH/bin/`. Therefore the lines

```
export PETSC_DIR=/home/user/petsc-2.3.3-p2/
export PETSC_ARCH=linux-gnu-c-debug
export PATH=$PETSC_DIR/externalpackages/mpich2-1.0.4p1/$PETSC_ARCH/bin/:$PATH
```

could appear in one of those files.

See Table 1 for a summary of the dependencies on external libraries, including those mentioned so far.

Installing PISM itself. At this point you have configured the environment which PISM needs. You are ready to build PISM itself, which is a much quicker procedure!

6. Get the latest source for PISM using the Subversion version control system:

```
svn co http://svn.gna.org/svn/pism/trunk pism
```

A directory called "`pism/`" will be created. Note that in the future when you enter that directory, `svn update` will update to the latest revision of PISM.[2]

7. Build PISM:[3]

```
cd pism
make
make install
```

Note that the "`make install`" puts executables, including `pismd`, `pismr`, `pismv`, and `pisms`, in the `pism/bin/` subdirectory and cleans up the junk from the build process. "`make install`" does *not* require root permission.

8. PISM executables can be run most easily by adding the directories `pism/bin/` and `pism/test/` to your `PATH`. The former directory contains the major PISM executables while the latter contains several useful scripts. For instance, this command can be done in the `bash` shell or in your `.bashrc` file:

```
export PATH=/home/user/pism/bin/:/home/user/pism/test/:$PATH
```

Quick tests of the installation. You are done with installation at this point. The next few items are recommended as they allow you to observe that PISM is functioning correctly.

10. Try a serial verification run of PISM:

```
pismv -test G -y 100
```

[2]Of course, after `svn update` you will `make` and `make install` to recompile and relink PISM.

[3]Please report any problems you meet at this build stage by sending us the output: `ffelb@uaf.edu`.

If you see some output and a final `Writing model state to file 'verify.nc' ... done` then PISM completed successfully. Note that at the end of this run you get measurements of the difference between the numerical result and the exact solution [10].

11. Try the MPI four processor version of the above run:

```
mpiexec -n 4 pismv -test G -y 100
```

This should work even if there is only one actual processor on your machine, in which case MPI will run multiple processes on the one processor, naturally. The reported errors should be very nearly the same as the serial run above, but the results should appear faster (if there really are four processors)!

12. Try a verification run on a finer vertical grid while watching the diagnostic views which use Xwindows:

```
pismv -test G -Mz 201 -y 2000 -d HTc
```

When using such diagnostic views and `mpiexec` the additional final option `-display :0` is sometimes required to enable MPI to use Xwindows:

```
mpiexec -n 2 pismv -test G -Mz 201 -y 2000 -d HTc -display :0
```

13. Run a verification test of the ice stream code:

```
pismv -test I -Mx 5 -My 401 -verbose
```

This runs a rather different part of the PISM code and then compares the numerical result to the exact solution appearing in [62].

14. Run a Python script for a basic suite of verifications:

```
verifynow.py
```

or, on an N processor machine,

```
verifynow.py -n N
```

If you would like us to confirm that PISM is working as expected please save the one page or so of output from this script and send it to us (`ffelb@uaf.edu`). See section 7 for more on PISM verification.

At this stage you can do the EISMINT II simplified geometry experiments and run some verification tests without further downloads. Subsection 9.1 is an example of using PISM to model the Greenland ice sheet using freely-downloadable data. Similarly one can model the Ross ice shelf using free data as described in subsection 9.2.

Naturally, setting up PISM to model real ice sheets will generally require techniques not be covered in this manual. First of all one usual needs to convert ice sheet data to NetCDF format so that it can be read by PISM. Actual modelling may also require the writing of additional source code. As PISM is written in C++, this means writing a derived class of the base class. (The base class `IceModel` is defined in the source file `pism/src/iceModel.hh`.) Use of PISM for real ice sheet modelling is something we welcome questions about, and will attempt to help with, but we will not pretend it is routine.

A final reminder with respect to installation: Let's assume you have checked out a copy of PISM using Subversion, as in step 6 above. You can update your copy of PISM to the latest version by `svn update` in the `pism/` directory. After doing so you will want to `make` and `make install` in order to rebuild the PISM executables using the updated source code files.

Have fun!

TABLE 1. Dependencies for PISM, listed alphabetically. These are needed to build a fully-functional PISM from source and to do all the examples in the manual.

Library /Program	**Site**	**Required?**	**Comment**
FFTW	`www.fftw.org`	*recommended*	if not present set `WITH_FFTW=0` for PISM build
GSL	`www.gnu.org/software/gsl`	*required*	
Matlab	`www.mathworks.com`	*optional*	only used for alternate display of results
MPI	`www-unix.mcs.anl.gov/mpi`	*required*	
NetCDF	`www.unidata.ucar.edu/software/netcdf`	*required*	
`numpy`	`numpy.scipy.org`	*recommended*	used in Python scripts in section 9
PETSc	`www-unix.mcs.anl.gov/petsc`	*required*	version $\geq$ 2.3.3-p2
Python	`python.org`	*required*	
`pycdf`	`pysclint.sourceforge.net/pycdf`	*recommended*	used in Python scripts in section 9
Subversion	`subversion.tigris.org`	*required*	

TABLE 2. Additional dependencies for PISM, listed alphabetically. *These are needed only for serious developers of PISM.*

Library /Program	**Site**	**Comment**
LaTeX	`www.latex-project.org`	only used for rebuilding manuals (both this User's Manual and the Reference Manual) from source
`doxygen`	`www.doxygen.org`	only used for rebuilding the Reference Manual and the Source Code Browser from the source code files
`ruby`	`www.ruby-lang.org`	only used when changing the NetCDF format for PISM output files

3. Getting started

3.1. **Running the EISMINT II tests.** PISM's purpose is the simulation of actual ice sheets. But actual ice sheet simulations require actual data.[4] For now, in order to avoid issues of data formats (and data flaws) in a first use of PISM, this section describes how to use PISM for experiment F in the EISMINT II simplified-geometry, thermomechanically-coupled ice sheet model intercomparison [50]. In this experiment one models an angularly-symmetric shallow, grounded ice sheet on a flat bed with relatively cold surface temperatures. Unfortunately one happens to be approximating an unstable equilibrium of the relevant (thermomechanically-coupled) partial differential equations, so one inevitably gets "spokes" in the basal temperature field [10, 51, 60].

In EISMINT II the prescribed grid has 60 subintervals in each direction, with each subinterval of length 25km. Thus the total width of the computational box is 1500 km in both x and y directions. However, PISM always allows choice of the grid in all three dimensions. A runtime option chooses the number of grid points in each direction; note that the number of points is one greater than the number of subintervals (grid spaces). The vertical grid is not prescribed in EISMINT II.

We choose the standard 25km grid in the horizontal and use 201 grid points in the vertical for a 25 m (equally-spaced) grid. Note that the computational box is 5000 m high by default for EISMINT II experiment F in PISM. Experiment F starts with zero ice, but the center thickness of the ice sheet grows to a peak near 5000 m before decaying to an equilibrium center thickness of about 4400 m.

In EISMINT II all runs are for 200,000 model years. Here we start with a short 2000 year run for a quicker illustration. The executable is "`pisms`", a name which has trailing "`s`" for the "simplifed geometry mode" of PISM:

```
$  pisms -eisII F -Mx 61 -My 61 -Mz 201 -y 2000
PISMS (simplified geometry mode)
initializing EISMINT II experiment F ...
  [computational box for ice: ( 1500.00 km) x ( 1500.00 km) x ( 5000.00 m)]
  [grid cell dimensions     : (   25.00 km) x (   25.00 km) x (   25.00 m)]
running EISMINT II experiment F ...
            YEAR (+     STEP[N$]):     VOL    AREA    MELTF     THICK0     TEMP0
$$$$$      0.000 (+  0.00000[0 ]):   0.000   0.000    0.000      0.000   223.150
$$vtf     60.000 (+ 60.00000[0m]):   0.017   0.628    0.000     30.000   223.150
$$vtf    120.000 (+ 60.00000[0m]):   0.034   0.628    0.000     60.000   223.538
$$vtf    180.000 (+ 60.00000[0m]):   0.051   0.628    0.000     90.000   223.869
...
$$vtf   1980.000 (+ 60.00000[0m]):   0.562   0.631    0.000    990.000   228.487
$$vtf   2000.000 (+ 20.00000[0e]):   0.568   0.631    0.000   1000.000   228.520
done with run ...
Writing model state to file 'simp_exper.nc' ... done.
```

[4]Actual ice sheet and ice shelf data *are* available on the web as part of the EISMINT intercomparison efforts. Section 9 is a tutorial on the use of PISM as a Greenland ice sheet or Ross ice shelf flow model using the EISMINT data sets.

This should have taken less than 30 seconds.

In a moment we will address the standard output information provided by PISM, as shown above, but for now we simply illustrate how to restart and complete the 200,000 year run. We see that the model state was stored in a NetCDF file with the `pisms` default name "`simp_exper.nc`". The next run will use a "`-o`" option to name the output file. Also, the above was a single processor run, but let's suppose we have a four processor machine. (The following should also work fine on a single processor machine by dropping the prefix "`mpiexec -n 4`".) Let's also run things in the background so we can continue to experiment:

```
$  mpiexec -n 4 pisms -eisII F -if simp_exper.nc -y 198000 \
  -o eisIIF200k >> eisIIF.out &
```

This run will take at least an hour on a four processor computer. The file `eisIIF200k.nc` (NetCDF format) will appear at the end. One can, however, view the redirected standard output by `less eisIIF.out` as the job is running in the background.[5]

If one wants a view of the model state in the midst of this (or any other PISM) run, one can send all running `pisms` processes a signal which causes PISM to write out the model state: `pkill -USR1 pisms`. The PISM model state is then saved in a NetCDF file with name `pism-`*year*`.nc` using the year at that time step. On the other hand, if one terminates the run with `pkill -TERM pisms` then the run will stop and save the model state using the `-o` specified name even though it will not be the state at the end of a completed run.[6] (See subsection 6.4 for a more complete description of how PISM catches signals.)

While the four process run continues in the background, let's view the model state during a few time steps by starting from the saved 2000 year state. We use PISM's "diagnostic viewers", which requires X Windows:

```
$  pisms -eisII F -if simp_exper.nc -y 200 -d HTt
PISMS (simplified geometry mode)
initializing from NetCDF format file  simp_exper.nc  ...
  [computational box for ice: ( 1500.00 km) x ( 1500.00 km) x ( 5000.00 m)]
  [grid cell dimensions     : (   25.00 km) x (   25.00 km) x (   25.00 m)]
running EISMINT II experiment F ...
           YEAR (+     STEP[N$]):     VOL    AREA    MELTF     THICK0     TEMP0
$$$$$   2000.000 (+  0.00000[0 ]):   0.568   0.631    0.000   1000.000   228.520
$$vtf   2060.000 (+ 60.00000[0m]):   0.585   0.631    0.000   1030.000   228.616
$$vtf   2120.000 (+ 60.00000[0m]):   0.602   0.631    0.000   1060.000   228.710
$$vtf   2180.000 (+ 60.00000[0m]):   0.619   0.631    0.000   1090.000   228.804
$$vtf   2200.000 (+ 20.00000[0e]):   0.625   0.631    0.000   1100.000   228.834
done with run ...
Writing model state to file 'simp_exper.nc' ... done.
```

[5] `top` is a convenient Linux tool to see processor usage during the run.

[6] Indeed, if the reader is impatient or has a single processor machine and doesn't want to wait more than an hour, the experiment F run can be terminated by `pkill -TERM pisms` after about 50,000 model years and the basic model result will be similar to that at 200,000 years.

Three figures should appear and be refreshed at each time step. One figure is a map-plane view of thickness, another is a map-plane view of the basal temperature in Kelvin, and third there is a graph of height above the bed versus temperature.

When the full 200,000 year run finishes, one can continue the run and watch its evolving state by

```
pisms -eisII F -if eisIIF200k.nc -d HTt
```

A result like that shown in figure 1 will appear.

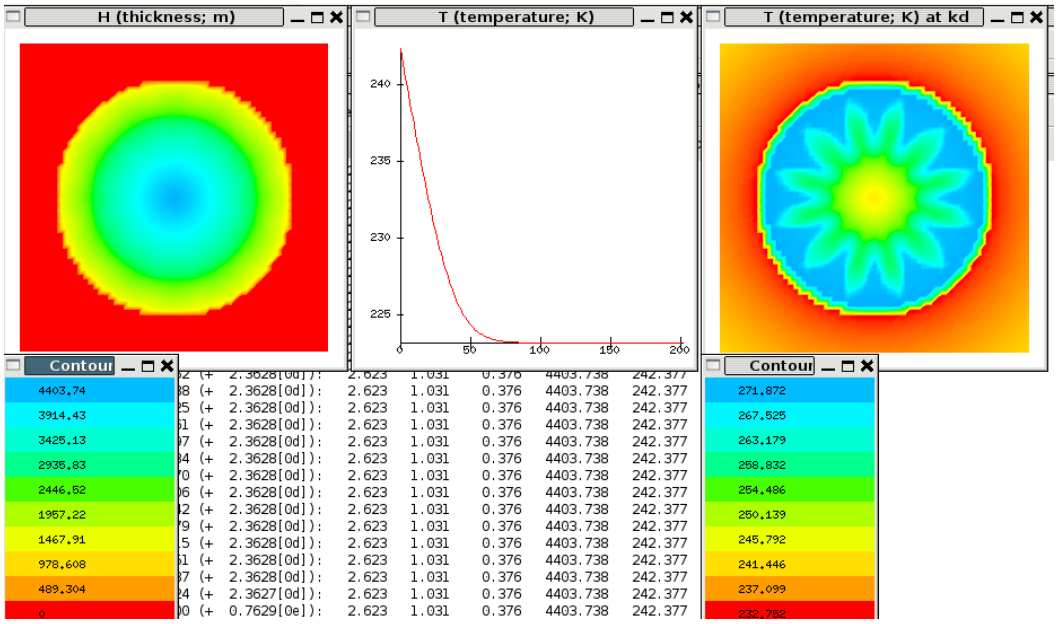


FIGURE 1. Diagnostic figures at the end of a 200,000 year EISMINT II experiment F run, showing the famous spokes.

At each time step PISM shows a summary of the model state using a few numbers. The format of the summary is

```
YEAR (+    STEP[N$]):     VOL    AREA    MELTF    THICKO    TEMPO
```

The first five columns are flags telling the user which quantities are being updated at that time step. A dollar sign appears if the quantity does not update. From the left the positions are: [`b$`] for bed elevation, [`vV$`] for velocity, [`g$`] for grain size, [`t$`] for temperature and age (which are always updated together), and [`f$`] for surface elevation. (That is, an "`f`" in the last column means a step of the flow or mass conservation equation has occurred. Regarding the two possible velocity flags, a lower case "`v`" indicates that the 3D velocity field has been updated, e.g. as needed for the advection of temperature, while uppercase "`V`" indicates that only the vertically-averaged velocity, and the associated diffusivity, has been updated.)

The time ("`YEAR`") and time step ("`STEP`") are in years.[7] A small whole number and a single character flag appear in square brackets after the time step, and these explain what part of the

[7]At the beginning of the EISMINT II experiment F run the ice has small thickness so the time step is 60 years because that is the default maximum time step. Later in the experiment F run, after 5000 years, for instance,

adaptive time-stepping scheme was used to determine the time step. For instance, "`m`" means that the step was the maximum allowed, "`e`" means that the time step was shortened to hit the end of the specified run, "`d`" means the step was determined by the diffusivity of the flow equation [10], and "`c`" means the CFL condition limited the time step [10, 45]. The small integer before this single character flag is related to the `-tempskip` mechanism; see sections A and 8 for further description and examples of this mechanism.

The next three columns in the summary report the volume of the ice in $10^6\,\mathrm{km}^3$, the area covered by the ice in $10^6\,\mathrm{km}^2$, and the basal melt fraction, that is, the fraction of the ice area where the basal homologous temperature is above 273.0 (i.e. slightly lower than the triple point). The next two columns "`THICK0`" and "`TEMP0`" are values at the center of the computational domain of the map plane, namely the thickness in meters and the basal absolute temperature in Kelvin. These five numbers are also the ones reported in the tables in [50], which is sometime useful for comparison. This summary of the model state can be made more verbose by using the option `-verbose`.

For more on the EISMINT II experiments see section 8.

3.2. **Visualizing the results.** There are two basic modes for visualizing the various quantities in PISM, namely using runtime viewers and visualizing output NetCDF files.

At runtime, various viewers can be specified by options of the form `-d` *letters* or `-dbig` *letters*; the latter type are fairly large. See Appendix C for the possible single letter names of each runtime viewer. These viewers are updated at each step and work under Xwindows. The format is limited by the style of PETSc viewers, but these viewers suffice for quick visualization and allow the monitoring of the run. Note that some diagnostic viewers show slices parallel to the bed. They are controlled by the option `-kd`. Those which show "soundings" are controlled by the options `-id`, `-jd`; see Appendix A.

When a PISM run is finished the state of the model is output in NetCDF format. The name can be specified by the option `-o`, so `-o foo` writes the NetCDF file `foo.nc`. NetCDF files can be viewed or modified with a variety of tools, some of which are mentioned in table 3.[8]

TABLE 3. Tools for viewing and modifying NetCDF files.

Tool	**Site**	**Function**
`ncdump`	*included with any NetCDF distribution*	dump as text file
`ncview`	`http://meteora.ucsd.edu/~pierce/ncview_home_page.html`	quick graphical view
`ncBrowse`	`www.epic.noaa.gov/java/ncBrowse/`	quick graphical view
`IDV`	`www.unidata.ucar.edu/software/idv/`	more complete visualization
MATLAB/netCDF interface	`www.marine.csiro.au/sw/matlab-netcdf.html`	read and write from MATLAB
`NCO` = the NetCDF Operators	`nco.sourceforge.net/`	sophisticated manipulations at command line

See `www.unidata.ucar.edu/software/netcdf/docs/software.html` for additional tools.

the time step is made smaller by the adaptive time-stepping mechanism in PISM. See subsection 6.3 for more on adaptive time-stepping.

[8]The PISM developers find `ncview` to be the fastest routine way to look at PISM output files.

3.3. **Evolution runs versus "diagnostic" runs.** The main goal of a numerical ice sheet model like PISM is to be a dynamical system which evolves over time as similarly as possible to the modelled ice sheet. Such a goal assumes one starts with the right initial conditions and that one has the right climate and other inputs at each time step (i.e. boundary conditions). Underlying an ice sheet model like PISM are evolution-in-time partial differential equations. The natural execution mode for a numerical ice sheet model is to take small time steps in approximating these differential equations; "small time steps" could mean several years, of course. We will describe this usual time-stepping behavior as an "evolution run."

Much of ice sheet, stream, and shelf modelling, however, requires "diagnostic" solution only of the partial differential equations which determine the velocity field. These are the balance of momentum equations for a slowly flowing fluid [17]. (As explained in the next section, the "shallow ice approximation" which underlies the EISMINT II experiments is only one of two forms of the balance of momentum equations solved by PISM. The other is the "shallow shelf approximation".) In a diagnostic computation of this type the temperature field and certain basal conditions, like the yield stress of the till for instance, are held fixed. In summary, the goal of a "diagnostic run" is to compute the velocity field; this will be the definition used from now on in this manual.

On the other hand, the NetCDF model state saved by PISM at the end of an evolution run does not contain the three-dimensional velocity field. Instead, it contains those variables which are needed to restart the run, especially the geometry (thickness and bed elevation) and the ice temperature field. For these reasons there is a separate executable `pismd` which only does a diagnostic computation of velocity. It saves the full three-dimensional velocity field but it does not do time-stepping.

For example using the quantities already computed in this section, try

```
pismd -if eisIIF200k.nc -o eisIIF200k_fullvels
```

The result of this command is a NetCDF file `eisIIF200k_fullvels.nc` which contains the full three-dimensional velocity field in the scalar NetCDF variables `u`, `v`, and `w`.

The velocity field saved by `pismd` is the one which would be computed at the *next* time step in an evolution run. That is, if we also run

```
pisms -eisII F -if eisIIF200k.nc -y 0.1 -o eisIIF200k_plus
```

then the horizontal velocity field computed in this single time step run is the same as that saved in `eisIIF200k_fullvels.nc`.[9] One can also force PISM to save the full velocity field at the end of a time-stepping run using the option `-full3Dout`. Either way, saving the full velocity field roughly doubles the size of the output model state NetCDF file.

Subsection 9.2 describes the use of `pismd` in modelling the Ross ice shelf. Indeed, "ice shelf modelling" has historically meant diagnositic and not evolution computations [41, 28].

[9]For this example, one can use a NetCDF Operator to check that the two saved NetCDF files really correspond to the same vertically-averaged horizontal velocity field; just do `ncdiff -v cbar` ...

4. Ice dynamics in PISM

4.1. **Two models of ice flow: slow versus fast, but always shallow.** PISM can numerically solve two significantly different sets of equations which determine the velocity field within the ice. In both cases the geometry of the ice, the temperature field within the ice, and the stress condition at the base of the ice are included into balance of momentum equations to determine the flow. But there are two different versions of the balance of momentum equations:

- the shallow ice approximation (SIA) [29], which describes ice flow as a function of the driving shear stresses of classical glaciology (i.e. in planes parallel to the geoid) [47], and
- the shallow shelf approximation (SSA) [66], which describes a membrane-type flow driven by longitudinal stress gradients [43, 40, 62].

PISM numerically solves both the SIA and SSA equations in parallel. The SIA equations are, as a rule, easier and quicker to numerically solve than the SSA. Certainly the SIA equations are easier to parallelize.

The SIA equations can confidently be applied to the grounded part of ice sheets for which the basal ice is frozen to the bedrock and the bed topography is relatively slowly-varying [17]. PISM can solve these equations with additional basal shear stress dependent sliding, but this has decreasing validity as the contribution of the basal sliding to the total flow velocity increases.

The SSA equations can confidently be applied to large floating ice shelves because such shelves have small depth-to-width ratio and because there is no shear stress at the base of the floating ice [43, 44].

Ice sheets have faster-flowing *grounded* parts, however. These are usually called "ice streams" or "outlet glaciers". One other other types of fast flow appear at the margin of, and even well into the interior of, the Greenland [36] and Antarctic [3] ice sheets. Describing these faster-flowing grounded parts of ice sheets requires something more than the SIA. Ultimately one might use the full Stokes equations which represent the balance of *all* tensor components [17]. Alternatively one might use a "higher-order" but still shallow stress balance [6].

Though they apply with greatest confidence to ice shelves, one may apply the SSA equations with additional basal resistance terms to grounded ice. This was justified in the context of the Siple Coast ice streams of Antarctica by MacAyeal [40, 27] using a linearly-viscous model for the underlying till. A free boundary problem with essentially the same (SSA) balance equations is the Schoof [62] model of ice streams emerging from plastic till failure under parts of an ice sheet; see also [63]. These are the models PISM uses to describe faster-flowing ice, namely the SSA with either linearly-viscous or plastic till.

As noted, both the SIA and SSA models are *shallow* approximations. That is, the partial differential equations in these two approximations come by (different) small-parameter arguments, based on a small depth-to-width ratio, from the Stokes equations of a non-Newtonian fluid. For this small-parameter argument in the SIA case see [17]. For the corresponding SSA argument, see the appendices of [62].[10]

[10]The references are, of course, the right place to examine the continuum equations and their physical motivation. This manual naturally avoids serious discussion of continuum models.

Within PISM, each grid point in the horizontal grid is either SIA *or* SSA. Each horizontal grid point corresponds to a whole vertical column of grid points in the three-dimensional grid, so it is perhaps better to say that each vertical column of ice centered at a horizontal grid point is either SIA *or* SSA. When ice is floating (according to the usual floatation criterion [66]), the horizontal grid point is automatically marked as SSA.

The actual numerical solution of the SSA for SSA-marked points is not done by default for the executables `pismr`, `pismd`, or `pisms`. The user must add the flag `-ssa`.

In any case, one must decide which parts of the grounded ice sheet are modelled by the SIA and which by the SSA. The rest of this section largely focuses on how the user makes this choice in practice when using PISM. One way is to specify a *mask*. Alternatively one can make a plasticity assumption about the till and determine the region of fast flow through the solution of a free boundary problem [62].

The transition from SIA to SSA flow is an internal boundary within the model domain. Along this boundary, the numerical schemes in PISM attempt to maintain these continuum facts:

- **i)** continuity of the *vertically-averaged* horizontal velocity field across the SIA–SSA transition,
- **ii)** conservation of mass *in the map plane* across the SIA–SSA transition, and
- **iii)** conservation of energy in three dimensions across the SIA–SSA transition.

Of course the statement "attempt to" maintain is important because PISM does not have exact representations of any continuous functions at all. It only "knows" their gridded numerical approximations. See section 10 of this manual, and the *PISM Reference Manual*, for additional comments and references on the continuum models and numerical schemes in PISM.

4.2. **The flow-type mask.** PISM always keeps track of a "flag" for flow-type at each point in the map-plane (two-dimensional) grid. This flow-type flag can potentially change at each time step. We call the array in which this flow-type flag is stored the "mask". There are four valid values for the mask (at this time), shown in table 4.

TABLE 4. Values for the PISM mask.

Value	Name	Meaning
1	`MASK_SHEET`	ice is grounded and the SIA is applied
2	`MASK_DRAGGING`	ice is grounded and the SSA is applied if option `-ssa` is given
3	`MASK_FLOATING`	ice is floating and the SSA is applied if option `-ssa` is given
7	`MASK_FLOATING_OCEAN0`	same as `MASK_FLOATING` but the point was ice free ocean at initialization of the model by `-bif`

4.3. **Schoof's plastic till free boundary problem for ice streams.** In [62] Christian Schoof proposes a model for plastic failure of the basal till under ice sheets and he describes an associated free boundary problem for the SSA. This is a model for emergent ice streams within a larger ice sheet and ice shelf system. It explains the existence of ice streams by the plastic failure of the till and the SSA approximation of the balance of momentum.

To demonstrate PISM's implementation of Schoof's model we describe an example based on EISMINT II experiment I. This experiment is documented in [49], but experiment I is

not discussed in the published intercomparison [50] because the results for experiment I were "straightforward results with little variability between groups". Experiment I is identical to experiment A except for having "trough" bed topography. As specified, experiment I has nothing to do with the SSA equations or plastic till failure, and indeed there is no basal motion at all. We use it to build an ice sheet which "ought" to have an ice stream.

The following run builds the ice sheet for experiment I for the first 10,000 model years, at which point it is of nearly maximum volume in the 200,000 year run specified in EISMINT II:

```
pisms -eisII I -Mx 61 -My 61 -Mz 201 -y 10000 -o eis2I10k
```

To view the resulting ice sheet one might do

```
pisms -eisII I -if eis2I10k.nc -d bHh
```

(and kill or Ctrl-C this unneeded run when one has seen enough). Thereby one sees the trough bed topography and its effect on the horizontal extent and thickness of the ice sheet. Also do

```
pisms -eisII I -if eis2I10k.nc -d 0cLT
```

to see the (horizontal) speed of the ice, effective thickness of the till water, and the basal temperature.

Note that a significant fraction of the base is at the pressure-melting temperature but that the effective thickness of the till water is zero. This is because `pisms -eisII` enforces zero stored melt water as a special simplification of PISM's default conservation of energy model for the EISMINT II tests [50].

On the other hand, the yield stress of the till in PISM's default plastic till model is tied to the effective thickness of the basal water. Therefore we continue the run but let PISM's default energy model build up a nonzero amount of basal water:

```
pisms -eisII I -if eis2I10k.nc -track_Hmelt -y 1000 -o eis2Iwmelt
```

When this last run finishes, an interesting view of the result is

```
pisms -eisII I -if eis2Iwmelt.nc -track_Hmelt -d 0HLT
```

In particular we see that the effective thickness of the till water has increased to its maximum of 2 m in much of the area where the base is at the pressure-melting temperature. In particular there is a layer of till water at the base of the ice in the trough part of the ice sheet. For comparison purposes we also continue with the next 1000 years of this run:

```
pisms -eisII I -if eis2Iwmelt.nc -track_Hmelt -d 0cqL -y 1000 -o eis2Isia
```

Now we turn on Schoof's plastic till ice stream model starting with the saved state `eis2Iwmelt.nc`:

```
pisms -eisII I -if eis2Iwmelt.nc -track_Hmelt -d 0cqL -y 1000 \
  -ssa -plastic -super -o eis2Iplastic
```

[THIS PRODUCES TOO LITTLE SLIDING BUT IS ROUGHLY THE RIGHT INVOCATION. PLAY WITH THE PARAMETERS IN THE BASAL YIELD STRESS MODEL?]

This kind of plastic till free boundary problem is also solved by test I in the verification suite. Here is an example run with some viewers:

```
pismv -test I -Mx 5 -My 500 -d ncqu -verbose
```

The boundary conditions are a uniform slab of ice on a bed with constant slope, but with a central low in the till yield stress. Thus the center of the slab slides, and thus the ice flows under the SSA equations, because of the plastic failure, while at the margins the base of the ice does

not slide. That is, we have a single simplified ice stream. But in this case we have the exact solution published in [62] with which to compare the diagnostically-computed velocities. This test, and convergence of the numerical solution under grid refinement, are addressed in section 7.

5. Initialization and inverse problems: "bootstrapping" PISM

[TO BE ADDED; cite [37] as representative inverse modelling results which could be incorporated in a forward model]

6. More on usage

6.1. **The PISM coordinate system and grid.** PISM does all simulations in a computational box which is rectangular in the PISM coordinates.

The coordinate system has horizontal coordinates x, y and a vertical coordinate z. The z coordinate is measured positive upward from the base of the ice and it is exactly opposite to the vector of gravity. The surface $z = 0$ is the base of the ice, however, and thus is usually not horizontal in the sense of being parallel to the geoid. The surface $z = 0$ is the base of the ice both when the ice is grounded and when the ice is floating.

Bed topography is, of course, allowed. In fact, when the ice is grounded, the true physical vertical coordinate z', namely the coordinate measure relative to a reference geoid, is given by $z' = z + b(x, y)$ where $b(x, y)$ is the bed topography. The surface $z' = h(x, y)$ is the surface of the ice. Thus in the grounded case the equation $h(x, y) = H(x, y) + b(x, y)$ applies if $H(x, y)$ is the thickness of the ice.

In the floating case, the physical vertical coordinate is $z' = z - (\rho_i/\rho_s)H(x, y)$ where ρ_i is the density of ice and ρ_s the density of sea water. Again $z = 0$ is the base of the ice, which is the surface $z' = -(\rho_i/\rho_s)H(x, y)$. The surface of the ice is $h(x, y) = (1 - \rho_i/\rho_s)H(x, y)$. All we know about the bed elevations is that they are below the base of the ice when the ice is floating. That is, the *floatation criterion* $-(\rho_i/\rho_s)H(x, y) > b(x, y)$ applies.

The computational box can extend downward into the bedrock. As $z = 0$ is the base of the ice, the bedrock corresponds to negative z values regardless of the sign of its true (i.e. z') elevation.

The extent of the computational box, along with its bedrock extension downward, is determined by four numbers `Lx`, `Ly`, `Lz`, and `Lbz`. The first two of these are half-widths and have units of kilometers when set by options or displayed. The last two are vertical distances in the ice and in the bedrock, respectively, and have units of meters. See the sketch in figure 2.

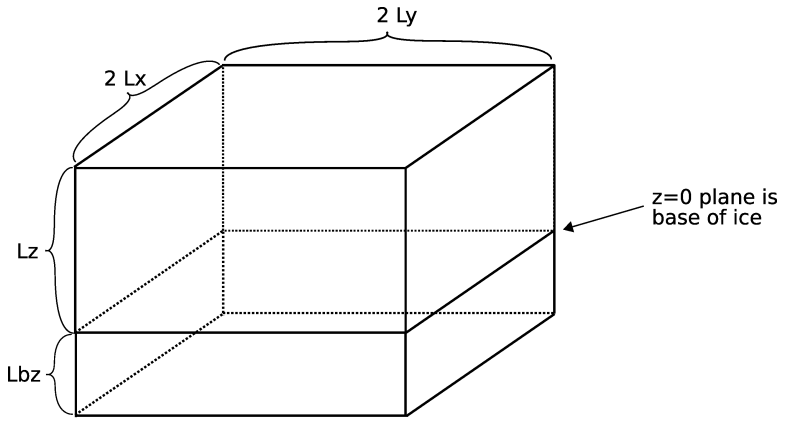


Figure 2. PISM's computational box.

The extent of the computational box for the ice is directly controlled by the options `-Lx`, `-Ly`, and `-Lz` as described in the Runtime options section. As noted `-Lx` and `-Ly` options should include values in kilometers while `-Lz` should be in meters.

The PISM grid covering the computational box is equally spaced in each of the three dimensions. Because of the bedrock extension, the grid of points is described by four numbers, namely the number of grid points in the x direction, the number in the y direction, the number in the z direction within the ice, and the number in the z direction within the bedrock. These are specified by options `-Mx`, `-My`, `-Mz`, and `-Mbz`, respectively, as described in the Runtime options section. The defaults for these four values are 61, 61, 31, and 1, respectively. Note that `Mx`, `My`, `Mz`, and `Mbz` all indicate the number of grid *points*. Therefore the number of grid *spaces* are, respectively, 60, 60, 30, and 0 (zero) in the default case. Note that the lowest grid point in a column of ice, that is the one at $z = 0$, coincides with the highest grid point in the bedrock. Also `Mbz` must always be at least one.

The distance `Lbz` is controlled by specifying the number `Mbz` of grid points in the bedrock, noting that the vertical spacing `dz` is the same within the ice and within the bedrock. To avoid conflicts, the distance `Lbz` cannot be set directly by the user. In particular, $\texttt{Lbz} = \texttt{dz}\,(\texttt{Mbz} - 1)$ while $\texttt{dz} = \texttt{Lz}/(\texttt{Mz} - 1)$, and so the distance `Lbz` into the bedrock is determined by setting `Lz`, `Mz`, and `Mbz`.

One is allowed to specify the grid when PISM is started *without* a pre-existing model state (i.e. as stored in a NetCDF input file output by PISM). For instance, a EISMINT II experiment F [50] run is

```
$  pisms -eisII F -Mx 61 -My 61 -Mz 101 -y 200000 -o foo
```

Note that PISM (i.e. the executable `pisms`) knows about the size of the computational box appropriate to each of the EISMINT II experiments.

If one initializes PISM from a saved model state then the input model state controls the parameters `Mx`, `My`, `Mz`, and `Mbz`. For instance, the command

```
$  pisms -eisII F -if foo.nc -Mz 201 -y 100
```

will give a warning that "`user option -Mz ignored; value read from file foo.nc`." To change the model grid one must explicitly "regrid", as described next.

6.2. **Regridding.** It is common to want to interpolate a coarse grid model state onto a finer grid or vice versa. For example, one might want to do the EISMINT II experiment F as above, producing `foo.nc`, but then interpolate both the ice thickness and the temperature onto a finer grid. Speaking conceptually, the idea in PISM is that one starts over from the beginning of EISMINT II experiment F on the finer grid, but one extracts the thickness and ice temperature stored in the coarse grid file and interpolates onto the finer grid before proceeding with the actual computation. The transfer from grid to grid is reasonably general—one can go from coarse to fine or vice versa in each dimension x, y, z—but the transfer must always be done by *interpolation* and never *extrapolation*. (An attempt to do the latter should always produce a PISM error.)

Such "regridding" is done using the `-regrid` and `-regrid_vars` commands as in this example:

```
$  pisms -eisII F -Mx 101 -My 101 -Mz 201 -y 1000 \
     -regrid foo.nc -regrid_vars HT -o bar
```

Note one specifies "`HT`" to indicate that the ice thickness and temperature values from the old grid should be interpolated onto the new grid. See table 5 for the regriddable variables. Note that one doesn't need to regrid the bed elevation, which is set identically zero as part of the

EISMINT II experiment F description, nor the ice surface elevation, which is computed as the bed elevation plus the ice thickness at each time step anyway.

A slightly different use of regridding occurs when "bootstrapping" as described in section 9. Here it is reasonable to have the sequence of commands, run on 8 processors,

```
mpiexec -n 8 pismr -bif_legacy init.nc -Mx 141 -My 141 -Mz 101 -Mbz 21 -gk -e 1.2 \
   -verbose -y 10 -o ant10yr_40km -of n

mpiexec -n 8 pismr -if ant10yr_40km.nc -gk -e 1.2 -no_mass -y 149990 \
   -o ant150k_40km -of n

mpiexec -n 8 pismr -bif_legacy init.nc -Mx 281 -My 281 -Mz 201 -Mbz 41 -gk -e 1.2 \
   -regrid ant150k_40km.nc -regrid_vars TBeL -verbose -y 10 -o ant150k_20km -of n
```

Here we bootstrap from an incomplete set of data in `init.nc` and smooth the surface for a short 10 year run on a coarseer 40km grid. Then we do a long simulation to complete 150,000 years on this coarse grid to approximate the temperature and age fields, assuming steady boundary conditions and fixed geometry. Finally we do a brief 10 year run with evolving geometry on a finer 20km grid, but we go back to the original data for the ice thickness; this last stage involves regridding temperature but not thickness, in particular.

TABLE 5. Regriddable variables. Use `-regrid_var` with given flag.

Flag	Variable	Comment
`a`	(net annual) accumulation	*climate data*; usually not regridded
`b`	bed elevation	
`B`	temperature in bedrock	
`e`	age of the ice	
`g`	geothermal flux	*climate data*; usually not regridded
`H`	thickness	
`L`	thickness of basal melt water (stored in till)	
`s`	ice surface temperature	*climate data*; usually not regridded
`T`	ice temperature	

6.3. **Understanding and controlling adaptive time-stepping.** Recall that at each time step we get a summary of the model state using a few numbers. The format of the summary is

```
YEAR (+    STEP[N$]):     VOL     AREA     MELTF     THICK0     TEMP0
```

Here we will explain what appears in the '`(+ STEP[N$])`' part of this summary.

`STEP` is the time step just taken by PISM, in model years. This time step is determined by a somewhat complicated adaptive mechanism. Note that PISM does each step explicitly when numerically approximating mass conservation in the map-plane. This requires that PISM have adaptive time-stepping for stability in the shallow ice approximation regions. Other issues like numerically approximating transport of temperature and age require adaptivity too [10].

Note that most of the time 'N' will be zero. The exception is when the option `-tempskip` is used. If `-tempskip` M is used, then `N` will be at most M, and will countdown the mass conservation steps when the adaptive scheme determines that a long temperature/age evolution time step, relative to the diffusity controlled time step for mass conservation, would be allowed. To see an example, do:

```
$  pismv -test G -Mx 141 -My 141 -Mz 51 -tempskip 4
```

Table 6 explains the meaning of the one character adaptive-timestepping flag '$'.

TABLE 6. Meaning of the adaptive time-stepping flag '$' in '(+ STEP[N$])'.

Flag	Active adaptive constraint
c	3D CFL for temperature/age advection [10]
d	diffusivity for SIA mass conservation [10]
e	end of prescribed run time
f	`-dt_force` set; generally option `-dt_force`, which overrides the adaptive scheme, should not be used
m	maximum allowed Δt applies; set with `-maxdt`
t	maximum Δt was *t*emporarily set by a derived class; e.g. see effect of deliverables `-time`n in `pisms -ismip H \time`n
u	2D CFL for mass conservation in SSA regions (where mass conservation is *u*pwinded)

6.4. **Using signals to control a running PISM model.** Ice sheet model runs sometimes take a long time and the state of the run needs checking. Sometimes they need to be stopped, but with the possibility of restarting. PISM implements these behaviors using "signals" from the POSIX standard. Such signals are included in Linux and most flavors of Unix. Table 7 below summarizes how PISM responds to signals.

Here is an example. Suppose we start a long verification run in the background, with standard out redirected into a file:

```
pismv -test G -Mz 101 -y 1e6 -o testGmillion >> log.txt &
```

This run gets a Unix process id ("`pid`"); we will need that number. (One can get the `pid` by using the `ps` or `pgrep` utilities.)

If we want to observe the run without stopping it we send the `USR1` signal:

```
kill -USR1 8920
```

where "8920" happened to be the `pid` of the running PISM process. It happens that we caught the run at year 31871.5 or so because a NetCDF file `pism-31871.495239.nc` is produced. This intermediate state file can be viewed by `ncview pism-31871.495239.nc`. Note also that in the standard out log file `log.txt` the lines

```
...
$vtf  31871.495 (+ 30.84701[0d]):   2.985   1.747    0.000   2997.637   272.236
Caught signal SIGUSR1: Writing intermediate file 'pism-31871.495239.nc'.
$vtf  31904.021 (+ 32.52611[0d]):   2.997   1.747    0.000   2997.641   272.235
...
```

appear.

Suppose, on the other hand, that the run needs to be stopped. One may use the interrupt "`kill -KILL 8920`" for this process, which is running in the background. (Foreground processes can be interrupted by Ctrl-C.) This brutal method, which a process cannot catch or block, stops the process but does not allow it time to save the model state, which is necessary to restart at the same place, or to inspect the model state more thoroughly. If one wants the possibility of restarting then one should used the `TERM` signal,

```
kill -TERM 8920
```

Then the PISM run is stopped and the lines

```
Caught signal SIGTERM: exiting early.
...
Writing model state to file 'testGmillion.nc' ... done
```

appear in the log file `log.txt`. In this case the NetCDF model state file `testGmillion.nc` appears; note this file has the original output name specified by the option `-o`. One can restart and finish the run by the command (for example)

```
pismv -test G -if testGmillion.nc -ye 1e6 -o testGmillion_finish >> log_cont.txt &
```

Finally we consider a multiple process (MPI) run of PISM. Here each of the processes must be sent the same signal at the same time. For example, consider the dialog

```
~/pism$ mpiexec -n 4 pismv -test C -y 100000 >> log.txt &
~/pism$ ps
PID TTY          TIME CMD
6761 pts/0    00:00:00 mpiexec
6762 pts/0    00:00:00 pismv
6763 pts/0    00:00:00 pismv
6764 pts/0    00:00:00 pismv
6765 pts/0    00:00:00 pismv
6770 pts/2    00:00:00 ps
~/pism$ kill -USR1 6762 6763 6764 6765
~/pism$ kill -TERM 6762 6763 6764 6765
```

Here the `kill` command sent the signal to all four of the running PISM processes simultaneously. The `kill -USR1` command caused the NetCDF file `pism-10852.037393.nc` to be written, while `kill -TERM` caused all four processes to end after (collectively, of course) writing `verify.nc`.

If, as in the above situation, one wants to send all `pismv` processes the `-TERM` signal then

```
pkill -TERM pismv
```

will have the same effect as `kill -TERM` followed by a list of all the `pid`s of `pismv` processes.

6.5. **Using the positive degree-day model.** By default the accumulation map in PISM is treated as net annual accumulation. If the input data is actually annual snow-fall (measured as ice-equivalent) then one must *compute* the net annual accumulation according to some mode of how much of the snow is melted in each model year. The mass conservation equation for the ice sheet requires the net annual accumulation, in any case. Also note that the surface temperature

TABLE 7. Signalling PISM. *pid* stands for the identifier of the running PISM process

Command	**Signal**	**PISM behavior**
`kill -KILL` *pid*	`SIGKILL`	terminate with extreme prejudice; PISM cannot catch it and no state is saved
`kill` -9 *pid*	(*same*)	(*same*)
Ctrl-C	(*same*)	same, but only for foreground processes
`kill -TERM` *pid*	`SIGTERM`	end process but save the last model state using the user-given `-o` name or the default name
`kill` -15 *pid*	(*same*)	(*same*)
`kill` *pid*	(*same*)	(*same*)
`pkill -TERM pismv`	(*same*)	(*same*, assuming one wants *all* running `pismv` processes to terminate and save)
`kill -USR1` *pid*	`SIGUSR1`	allow process to continue but save model state at current time using name `pism-`*year*`.nc`
`pkill -USR1 pismv`	(*same*)	(as above for `pkill`)

map in PISM is the mean annual surface temperature, so without an additional model or input there is no yearly temperature cycle.

A reasonable method for computing the melting of snowfall, optionally used by PISM, first assumes a sinusoidal temperature cycle over the course of the year. The amount of summer warming is the difference between the peak of this temperature cycle and its mean. The default temperature cycle has a constant amount of summer warming. This constant is specified by `-pdd_summer_warming`. See section 9 for an alternative model. Note that the mean temperature is grid-point dependent, as specified by the input surface temperature map (data), so the cycle is different at each map-plane grid location, though the amplitude is the same at each grid point in the default case. Also, note that, by default, the peak of the cycle occurs on August 1, but we do not believe that the specific date for "mid-summer" matters detectably to whole ice sheet dynamics or simulations thereof.

The part of this cycle above 0°C is converted to positive degree days. This number of positive degree days is multiplied by a coefficient (set by `-pdd_factor_snow`) to compute the amount of snow melted. Of this melted snow, a fraction (`-pdd_refreeze`) is kept as ice. This ice, plus all unmelted snow (measured as ice-equivalent) is applied as accumulation, unless the number of positive degree days exceeds that required to melt all of the snow-fall in the year. In this latter case in which there are excess positive degree days available for melting, the number of excess positive degree days is multiplied by a coefficient (`-pdd_factor_ice`) to compute how much ice is melted, so that net ablation occurs (at the given point).

As an additional feature one may add "white noise" to the temperature cycle. More precisely, a normally-distributed, mean zero random temperature increment is added (or subtracted) from the temperature for each day. These increments are independent over the days of the year, though of course we only have pseudo-randomness ..., but they are the same over the

whole sheet. Their standard deviation is controlled by `-pdd_std_dev`. If one wants runs with repeatable randomness, add the option `-pdd_repeatable`.

As an example of this mechanism, first do

```
pisms -eisII A -Mx 61 -My 61 -Mz 101 -y 8000 -o foo
```

to get an ice sheet which is large enough to have spread significantly into the ablation zone. Next observe how it behaves in a warmer "climate" *without* the positive degree day model; note that the peak surface temperature in EISMINT II experiment B is 5.0 degrees warmer than that in experiment A [50]:

```
pisms -eisII B -if foo.nc -y 40
```

In this 40 year run the ice sheet grows a bit. That is, the volume grows and the central thickness ("`THICK0`") grows.

Now observe how it the same saved ice sheet `foo.nc` behaves with the default positive degree day model (which is not a part of EISMINT II, of course):

```
pisms -eisII B -if foo.nc -y 40 -pdd
```

The central thickness still grows, by nearly the same amount, but of course the center is in the accumlation zone and is not strongly affected by melting. In this case with the positive degree day model the volume actually shrinks, however.

Note that last run is the same as

```
pisms -eisII B -if foo.nc -y 40 -pdd_summer_warming 15.0 -pdd_factor_snow 0.003 \
  -pdd_factor_ice 0.008 -pdd_refreeze 0.6 -pdd_std_dev 0.0
```

and that any of these options can be adjusted.

A positive degree day model specific to the Greenland ice sheet is used in section 9.

[THE IDEAS IN [13] ARE WORTHWHILE AND DESERVE IMPLEMENTATION. WITH THESE IDEAS THERE WILL BE NO RESTRICTION TO HITTING THE INTEGER YEARS. BUT THE TRUE RANDOMNESS WILL BE GONE; IT MAY BE WORTH KEEPING THE OLD VERSTION AS AN OPTION.]

7. Verification

> Two types of errors may be distinguished: modelling errors and numerical errors. Modelling errors arise from not solving the right equations. Numerical errors result from not solving the equations right. The assessment of modelling errors is *validation*, whereas the assessment of numerical errors is called *verification* ... Validation makes sense only after verification, otherwise agreement between measured and computed results may well be fortuitous.
>
> P. Wesseling, (2001) *Principles of Computational Fluid Dynamics*, pp. 560–561 [67]

Ideas. "Verification" is a crucial task for a code as complicated as PISM. It is the exclusively mathematical and numerical task of checking that the predictions of the numerical code are close to the predictions of the continuum model (the one which the numerical code claims to approximate). In particular, one compares exact solutions of the continuum model, in circumstance in which they are available, to the numerical approximations of those solutions.

See [12] and [10] for discussion of verification issues for the isothermal and thermomechanically coupled shallow ice approximation (SIA), respectively, and for exact solutions to these models. See [62] for an exact solution to the SSA equations for ice streams using a plastic till assumption. [58] gives a broader discussion of verification and validation throughout computational fluid dynamics.

In PISM there is a separate executable `pismv` which is used for verification. The numerical code which is verified by `pismv` is, however, exactly the same lines of code in exactly the same source files as is run by the non-verification executables `pismr`, `pisms`, `pgrn`, etc. (In technical terms, `pismv` runs a derived class of the core class `IceModel`; all PISM executables run the core class.)

Table 8 summarizes the many exact solutions contained within PISM. Note that all of these exact solutions, except tests A and E, are solutions of free boundary problems for partial differential equations. Table 9 shows how to run each of them for an example grid, namely one with relatively quick execution time.

Refinement. To meaningfully verify a numerical code one must go down a grid refinement path and measure error for each grid. By "a refinement path" we mean the specification of a sequence of grid cell sizes (e.g. a sequence of triples $(\Delta x, \Delta y, \Delta t)$ in the two spatial and one time dimension case) which decrease toward the (unreachable) infinite refinement limit $(\Delta x, \Delta y, \Delta t) = (0, 0, 0)$. By "measuring the error for each grid" we will concretely mean computing two norms, namely both the maximum absolute error anywhere on the grid, relative to the exact solution, and computing the average error on the grid. (These are essentially the L^1 and L^∞ norms and bracketing cases in the largest family of common norms [54].)

The goal is not to see that the error is zero at any stage on the refinement path, or even that the error is small in a predetermined absolute sense, generally. Rather the goal is to see that the error is trending toward zero, and to measure the rate at which decays. See [12, 10, 58, 67].

For an example of a refinement path, consider the runs

TABLE 8. Exact solutions for verification.

Test	Continuum model tested	Comments	Reference
A	isothermal SIA (mass conservation), steady, flat bed, constant accumulation		[12]
B	isothermal SIA, flat bed, zero accum	similarity soln	[12]
C	isothermal SIA, flat bed, growing accum	similarity soln	[12]
D	isothermal SIA, flat bed, oscillating accum	compensatory accum	[12]
E	isothermal SIA; as A but with sliding in a sector	compensatory accum	[12]
F	thermomechanically coupled SIA (mass and energy cons.), steady, flat bed	compensatory accum and comp heating	[9, 10]
G	thermomechanically coupled SIA; as F but with oscillating accumulation	ditto	[9, 10]
H	bed deformation coupled with isothermal SIA	joined similarity	[11]
I	stream velocity computation using SSA (plastic till)		[62]
J	shelf velocity computation using SSA		[PREPRINT]
K	pure conduction in ice and bedrock		[PREPRINT]
L	isothermal SIA, steady, non-flat bed	numerical ODE soln	[8]

TABLE 9. Running PISM to verify using the exact solutions listed in table 8.

Test	Example invocation
A	`pismv -test A -Mx 61 -My 61 -Mz 11 -y 25000`
B	`pismv -test B -Mx 61 -My 61 -Mz 11 -ys 422.45 -y 25000`
C	`pismv -test C -Mx 61 -My 61 -Mz 11 -y 15208.0`
D	`pismv -test D -Mx 61 -My 61 -Mz 11 -y 25000`
E	`pismv -test E -Mx 61 -My 61 -Mz 11 -y 25000`
F	`pismv -test F -Mx 61 -My 61 -Mz 61 -y 25000`
G	`pismv -test G -Mx 61 -My 61 -Mz 61 -y 25000`
H	`pismv -test H -Mx 61 -My 61 -Mz 11 -y 40034 -bed_def_iso`
I	`pismv -test I -Mx 5 -My 500 -ssa_rtol 1e-6 -ksp_rtol 1e-11`
J	`pismv -test J -Mx 60 -My 60 -Mz 11 -ksp_rtol 1e-12`
K	`pismv -test K -Mx 6 -My 6 -Mz 401 -Mbz 101 -y 130000`
L	`pismv -test L -Mx 61 -My 61 -Mz 31 -y 25000`

```
pismv -test B -ys 422.45 -y 25000 -Mx 31 -My 31 -Mz 11
pismv -test B -ys 422.45 -y 25000 -Mx 61 -My 61 -Mz 11
pismv -test B -ys 422.45 -y 25000 -Mx 121 -My 121 -Mz 11
pismv -test B -ys 422.45 -y 25000 -Mx 241 -My 241 -Mz 11
```

These verify the basic function of the isothermal shallow ice approximation components of PISM in the case of no accumulation. Note that one specifies the number of grid spaces when running PISM, but this is equivalent to specifying the grid cell sizes if the overall dimensions of the computational box is fixed; see subsection 6.1. The exact solution in used here is the

Halfar similarity solution [23]. The refinement path is the sequence of triples $(\Delta x, \Delta y, \Delta)$ with $\Delta x = \Delta y = 80, 40, 20, 10$ and where Δt is determined adaptively by a stability criterion (see subsection 6.3). Note that the vertical grid spacing Δz is fixed because this test is isothermal and no dependence of the error is expected from changing Δz.

The data produced by the above four runs appears in figures 7, 8, 9, and 10 of [12]. We see there that the isothermal mass conservation scheme does a reasonable job of approximating the evolving surface. (But we also acknowledge the fact that future improvements in the numerical scheme can make the error decrease more quickly or slowly.)

For thermocoupled tests one refines in three dimensions. For example, the runs

```
pismv -test G -maxdt 10.0 -y 25000 -Mx 61 -My 61 -Mz 61
pismv -test G -maxdt 10.0 -y 25000 -Mx 91 -My 91 -Mz 91
pismv -test G -maxdt 10.0 -y 25000 -Mx 121 -My 121 -Mz 121
pismv -test G -maxdt 10.0 -y 25000 -Mx 181 -My 181 -Mz 181
pismv -test G -maxdt 10.0 -y 25000 -Mx 241 -My 241 -Mz 241
pismv -test G -maxdt 10.0 -y 25000 -Mx 361 -My 361 -Mz 361
```

produced figures 13, 14, and 15 of [10]. (The last couple of these runs required a supercomputer! The $361 \times 361 \times 361$ run involves more than 100 million unknowns, updated at each of millions of time steps.)

`verifynow.py`. This is the Python script in subdirectory `pism/test/` which organizes the process of verifying PISM. By default it runs Tests C, G, and I, because these tests exercise most of the numerical methods in PISM, and thus suffice for basic verification.

Three example usages are

- "`verifynow.py`" without options will use one processor and do three levels of refinement; this command is equivalent to `verifynow.py -n 1 -l 3 -t CGIJ`
- "`verifynow.py -n 8 -l 5 -t J --prefix=bin/ --mpido=mpirun/`" will use `mpirun -np 8 bin/pismv` as the command and do five levels (the maximum) of refinement only on test J
- "`verifynow.py -n 40 -l 5 -t ABCDEFGIJKL`" will use lots of processors to do all possible verification as managed by `verifynow.py`; don't run this unless you have a big computer and you are prepared to wait

In fact, `verifynow.py` takes the following options. They have both a short form and a long form, *a la* GNU; the default is in brackets:

`-l`, `--levels=` [*3*]: specifies number of levels of refinement; $1, 2, 3, 4, 5$ are allowed values

`-m`, `--mpido=` [*`mpiexec`*]: how to run PISM as an MPI program

`-n`, `--nproc=` [*1*]: specify number of processors to use

`-p`, `--prefix=` []: where the PISM executable `pismv` is located

`-t`, `--tests=` [*CGIJ*]: which tests to run

Here is the result of three levels of refinement for tests C, E, G, I, and L. Note that the errors generally decay, with the notable exception of the surface values of the vertical velocity in test G; compare [61]. The exact meaning of the various errors is currently only documented in

the source files `pism/src/iceCompModel.cc` and `pism/src/iCMthermo.cc`. Note that timing information is given, so performance, including parallel performance, can be assessed along with accuracy.

FIXME: following is out of date; errors for surface values of w DO now decrease

```
$ verifynow.py -n 2 -l 3 -t CEGIL
 VERIFYNOW using 2 processor(s) and 3 level(s) of refinement
 ++++ verifying isothermal SIA w moving margin using test C ++++
 (Mx=My=41,61,81,101,121 corresponds to dx=dy=50,33.3,25,20,16 km)
 trying "mpiexec -np 2 pismv -test C -Mx 41 -My 41 -Mz 31 -y 15208.0 -verbose 1"
 finished in  9.2709 seconds; reported numerical errors as follows:
  |NUMERICAL ERRORS evaluated at final time (relative to exact solution):
  |geometry  :    prcntVOL        maxH         avH   relmaxETA      centerH
  |               0.718900  250.003494   12.015517    0.019159    10.040006
 trying "mpiexec -np 2 pismv -test C -Mx 61 -My 61 -Mz 31 -y 15208.0 -verbose 1"
 finished in 41.2792 seconds; reported numerical errors as follows:
  |NUMERICAL ERRORS evaluated at final time (relative to exact solution):
  |geometry  :    prcntVOL        maxH         avH   relmaxETA      centerH
  |               0.126767  224.348860    7.129075    0.012296     4.401402
 trying "mpiexec -np 2 pismv -test C -Mx 81 -My 81 -Mz 31 -y 15208.0 -verbose 1"
 finished in 122.7838 seconds; reported numerical errors as follows:
  |NUMERICAL ERRORS evaluated at final time (relative to exact solution):
  |geometry  :    prcntVOL        maxH         avH   relmaxETA      centerH
  |               0.197003  194.241335    4.131559    0.008131     2.126740
 ++++ verifying isothermal SIA w sliding using test E ++++
 (Mx=My=31,41,61,81,121 corresponds to dx=dy=53.3,40,26.7,20,13.3 km)
 trying "mpiexec -np 2 pismv -test E -Mx 31 -My 31 -Mz 31 -y 25000.0 -verbose 1"
 finished in 85.2384 seconds; reported numerical errors as follows:
  |NUMERICAL ERRORS evaluated at final time (relative to exact solution):
  |geometry  :    prcntVOL        maxH         avH   relmaxETA      centerH
  |               6.144464  920.922943   85.952669    0.078342    47.073721
  |base vels :  maxvector   avvector  prcntavvec     maxub     maxvb
  |               12.4115    0.91501     1.80232   10.0374    7.5887
 trying "mpiexec -np 2 pismv -test E -Mx 41 -My 41 -Mz 31 -y 25000.0 -verbose 1"
 finished in 193.6552 seconds; reported numerical errors as follows:
  |NUMERICAL ERRORS evaluated at final time (relative to exact solution):
  |geometry  :    prcntVOL        maxH         avH   relmaxETA      centerH
  |               4.481193  726.112323   63.817098    0.059375    36.149340
  |base vels :  maxvector   avvector  prcntavvec     maxub     maxvb
  |                8.7523    0.67766     1.33480    8.6051    5.2778
 trying "mpiexec -np 2 pismv -test E -Mx 61 -My 61 -Mz 31 -y 25000.0 -verbose 1"
 finished in 893.9928 seconds; reported numerical errors as follows:
  |NUMERICAL ERRORS evaluated at final time (relative to exact solution):
  |geometry  :    prcntVOL        maxH         avH   relmaxETA      centerH
  |               2.804588  636.954964   40.564555    0.040141    21.534520
  |base vels :  maxvector   avvector  prcntavvec     maxub     maxvb
  |                7.1069    0.44876     0.88394    6.0500    4.3875
 ++++ verifying plastic till ice stream using test I ++++
 (My=49,193,769,3073,12289 corresponds to dy=5000,1250,312.5,78.13,19.53 m)
 trying "mpiexec -np 2 pismv -test I -My 49 -Mx 5 -mv_rtol 5e-07 -ksp_rtol 1e-12 -verbose 1"
 finished in  1.1180 seconds; reported numerical errors as follows:
  |NUMERICAL ERRORS in velocity relative to exact solution:
  |      maxvector   avvector  prcntavvec      maxu      maxv       avu       avv
  |        23.3329    7.69421     0.98956   23.3329    0.0000    7.6942    0.0000
 trying "mpiexec -np 2 pismv -test I -My 193 -Mx 5 -mv_rtol 5e-07 -ksp_rtol 1e-12 -verbose 1"
```

```
finished in  3.2881 seconds; reported numerical errors as follows:
 |NUMERICAL ERRORS in velocity relative to exact solution:
 |      maxvector   avvector  prcntavvec      maxu      maxv       avu       avv
 |         1.3246    0.44147     0.05678    1.3246    0.0000    0.4415    0.0000
trying "mpiexec -np 2 pismv -test I -My 769 -Mx 5 -mv_rtol 5e-07 -ksp_rtol 1e-12 -verbose 1"
finished in 12.0152 seconds; reported numerical errors as follows:
 |NUMERICAL ERRORS in velocity relative to exact solution:
 |      maxvector   avvector  prcntavvec      maxu      maxv       avu       avv
 |         0.0923    0.03117     0.00401    0.0923    0.0000    0.0312    0.0000
++++ verifying isothermal SIA w non-flat bed using test L ++++
(Mx=My=31,61,91,121,181 corresponds to dx=dy=60,30,20,15,10 km)
trying "mpiexec -np 2 pismv -test L -Mx 31 -My 31 -Mz 31 -y 25000.0 -verbose 1"
finished in 13.2488 seconds; reported numerical errors as follows:
 |NUMERICAL ERRORS evaluated at final time (relative to exact solution):
 |geometry  :    prcntVOL        maxH         avH   relmaxETA      centerH
 |               0.152375  381.614581   15.555426    0.006689     6.533219
trying "mpiexec -np 2 pismv -test L -Mx 61 -My 61 -Mz 31 -y 25000.0 -verbose 1"
finished in 169.4863 seconds; reported numerical errors as follows:
 |NUMERICAL ERRORS evaluated at final time (relative to exact solution):
 |geometry  :    prcntVOL        maxH         avH   relmaxETA      centerH
 |               0.206286  359.083724    4.984371    0.002643     1.386802
trying "mpiexec -np 2 pismv -test L -Mx 91 -My 91 -Mz 31 -y 25000.0 -verbose 1"
finished in 821.1084 seconds; reported numerical errors as follows:
 |NUMERICAL ERRORS evaluated at final time (relative to exact solution):
 |geometry  :    prcntVOL        maxH         avH   relmaxETA      centerH
 |               0.159108  306.654292    3.789676    0.001773     1.542505
++++ verifying thermocoupled SIA w variable accum using test G ++++
(Mx=My=Mz=61,91,121,181,241 corresponds to dx=dy=30,20,15,10,7.5 km
and dz=66.7,44.4,33.3,22.2,16.7 m)
trying "mpiexec -np 2 pismv -test G -Mx 61 -My 61 -Mz 61 -y 25000.0 -verbose 1"
finished in 120.9870 seconds; reported numerical errors as follows:
 |NUMERICAL ERRORS evaluated at final time (relative to exact solution):
 |geometry  :    prcntVOL        maxH         avH   relmaxETA      centerH
 |               2.107584   64.364190   19.550635    0.017980     2.822391
 |temp      :        maxT         avT    basemaxT     baseavT  basecenterT
 |               1.920759    1.050554    1.533177    0.471904     0.069055
 |Sigma (3D):         max          av
 |               0.051570    0.002843
 |surf vels :     maxUvec      avUvec        maxW         avW
 |               0.263333    0.070287    0.022908    0.002659
trying "mpiexec -np 2 pismv -test G -Mx 91 -My 91 -Mz 91 -y 25000.0 -verbose 1"
finished in 882.4219 seconds; reported numerical errors as follows:
 |NUMERICAL ERRORS evaluated at final time (relative to exact solution):
 |geometry  :    prcntVOL        maxH         avH   relmaxETA      centerH
 |               0.950808   29.790284    9.032937    0.007974     0.009471
 |temp      :        maxT         avT    basemaxT     baseavT  basecenterT
 |               1.108004    0.556731    0.962611    0.238721     0.025394
 |Sigma (3D):         max          av
 |               0.025978    0.001190
 |surf vels :     maxUvec      avUvec        maxW         avW
 |               0.129822    0.032835    0.035400    0.004319
trying "mpiexec -np 2 pismv -test G -Mx 121 -My 121 -Mz 121 -y 25000.0 -verbose 1"
finished in 3634.9651 seconds; reported numerical errors as follows:
 |NUMERICAL ERRORS evaluated at final time (relative to exact solution):
 |geometry  :    prcntVOL        maxH         avH   relmaxETA      centerH
```

```
|               0.520703   28.348163    4.989002    0.004474     0.806247
|temp      :        maxT         avT     basemaxT     baseavT  basecenterT
|               0.810869    0.360046    0.781534    0.141099     0.009927
|Sigma (3D):         max          av
|               0.061469    0.000725
|surf vels :     maxUvec      avUvec        maxW         avW
|               0.089217    0.020179    0.046493    0.005915
```

8. Simplified geometry experiments

8.1. **Historical note.** There have been at least three stages of ice sheet model intercomparisons based on simplified geometry experiments since the early 1990s.

EISMINT I [31] was the first of these and involved only the isothermal shallow ice approximation (SIA). ("EISMINT" stands for European Ice Sheet Modeling INiTiative.) Both fixed margin and moving margin experiments were performed in EISMINT I, and various conclusions were drawn about the several numerical schemes used in the intercomparison.

EISMINT I is, however, superceded by *verification* using the full variety of exact solutions to the isothermal SIA. The reasons why EISMINT I can be fully replaced by verification are described in [12]. The "rediscovery", since EISMINT I, of the very useful Halfar similarity solution with zero accumulation [23] is a particular reason why the "moving margin" experiment in EISMINT I is, roughly speaking, irrelevant. For this reason there has been no attempt to support the EISMINT I experiments in PISM.

EISMINT II [50] was both a more significant and more controversial intercomparison. It pointed out interesting and surprising properties of the thermocoupled SIA. Here is not the place for a discussion of the interpretations which have followed the EISMINT II results, but references [10, 24, 25, 51, 60] each interpret the EISMINT II experiments and/or describe attempts to add more complete physical models to "fix" the perceived shortfalls of ice sheet models (as evidenced by their behavior on EISMINT II experiments). PISM has built-in support for all of the published and unpublished EISMINT II experiments; these are described in the next subsection.

The ISMIP round of intercomparisons are ongoing at the time of this writing (2007); "ISMIP" stands for Ice Sheet Model Intercomparison Project. At this time there are two components of ISMIP partially completed, namely HOM = Higher Order Models and HEINO = Heinrich Event INtercOmparison [22].

Of these ISMIP experiments, PISM currently only supports HEINO, but the results from PISM for HEINO are not regarded by the PISM authors as meaningful. (ISMIP-HEINO support in PISM is an *un*documented feature.) We believe the continuum problem described by HEINO to not be easily approximate-able because of a (large) discontinuous jump in the basal velocity field. This is a problem regardless of the details of the numerical schemes or even (roughly speaking) of the shallowness of the continuum model.

As of August 2007, the PISM developers plan to participate in the upcoming ISMIP-POLICE and Marine Ice Sheet Intercomparison exercises.

8.2. **EISMINT II in PISM.** There are seven experiments described in the published EISMINT II writeup [50]. They are labeled A, B, C, D, F, G, and H. As specified in the writeup, the common features of all of these experiments are:

- runs are of 200,000 years, with no prescribed time step;
- runs are on a prescribed 61×61 horizontal grid (but, as usual, the grid and the length of the run are command line options in PISM so PISM can easily run the same experiments on a refined grid);
- the boundary conditions always have angular symmetry around the center of the grid;

- the bed is always flat and does not move (so the effects of isostasy were ignored throughout);
- the temperature in the bedrock is not modelled;
- only shallow ice approximation physics is included;
- the change in the temperature of ice is described by the shallow approximation of the conservation of energy [17];
- thermomechanical coupling is included, both because of the temperature dependence of the softness of the ice, *and* through the strain-heating (dissipation-heating) term in the conservation of energy equation;
- the ice is *cold* and not *polythermal* [19]; and finally
- though basal melt rates may be computed diagnostically, they do not contribute to the dynamics of the ice sheet.

The experiments differ from each other in their various combinations of surface temperature and accumulation parameterizations. Also, experiments H and G involve basal sliding, while the others don't. Four experiments start with zero ice (A,F,G,H), while the other experiments (B,C,D) start from the final state of experiment A.

In addition to the seven experiments published in [50], there were an additional five experiments described in the EISMINT II intercomparison description [49], labeled E, I, J, K, and L. These experiments share most features, itemized above, but with the following differences. Experiment E is the same as experiment A except that the peak of the accumulation, and also the low point of the surface temperature, are shifted by 100 km in both x and y directions; also experiment E starts with the final state of experiment A. Experiments I and J are similar to experiment A but with non-flat topography. Experiments K and L are similar to experiment C but with non-flat topography. See table 10 for how to run these experiments.

The vertical grid is not specified in the EISMINT II writeup. It seems that good simulation of the complex thermomechanically coupled conditions near the base of the ice requires relatively fine resolution there. Because PISM (currently) has an equally-spaced grid, we recommend the use of about 200 vertical levels. Thus a reasonable experiment A run on one processor is

```
pisms -eisII A -Mx 61 -My 61 -Mz 201 -y 200000 -o eisIIA
```

The final state `eisIIA.nc` can, of course, be viewed using `ncview`.

Table 10 shows how each of the EISMINT II experiments can be done in PISM.

The EISMINT II experiments can be run with various modifications of the default settings. Of course the grid can be refined. For instance, a twice as fine grid in the horizontal is "`-Mx 121 -My 121`". Table 11 lists some optional settings which are particular to the EISMINT II experiments. With the exception of "`-Lz`", these options will only work if option "`-eisII ?`" is also set.

Note that in PISM the height `Lz` of the computational box is fixed at the beginning of the run. On the other hand, changing the boundary conditions of the flow, as for instance by setting option `-Mmax` to a larger than default value (see table 11), may cause the ice sheet to thicken above the `Lz` height. If the ice grows above the height of the computational box then a "`Vertical grid exceeded!`" or "`thickness overflow in SIA velocity: ks>Mz!`" error occurs. This can be fixed by remedied by restarting with a larger value for option `-Lz`.

TABLE 10. Running the EISMINT II experiments in PISM; the command is "`pisms`" plus the given options.

Command: "`pisms`" +	**Relation to experiment A**
`-eisII A -Mx 61 -My 61 -Mz 201 -y 2e5 -o eisIIA`	
`-eisII B -if eisIIA.nc -y 2e5 -o eisIIB`	warmer
`-eisII C -if eisIIA.nc -y 2e5 -o eisIIC`	less snow (lower accumulation)
`-eisII D -if eisIIA.nc -y 2e5 -o eisIID`	only smaller area of accumulation
`-eisII F -Mx 61 -My 61 -Mz 201 -y 200000 -o eisIIF`	colder; famous spokes [10]
`-eisII G -Mx 61 -My 61 -Mz 201 -y 200000 -o eisIIG`	sliding (regardless of temperature)
`-eisII H -Mx 61 -My 61 -Mz 201 -y 200000 -o eisIIH`	melt-temperature activated sliding
`-eisII E -if eisIIA.nc -y 2e5 -o eisIIE`	shifted accumulation/temperature maps
`-eisII I -Mx 61 -My 61 -Mz 201 -y 2e5 -o eisIII`	trough topography
`-eisII J -if eisIII -y 2e5 -o eisIIJ`	trough topography and less snow
`-eisII K -Mx 61 -My 61 -Mz 201 -y 2e5 -o eisIIK`	mound topography
`-eisII L -if eisIIK -y 2e5 -o eisIIL`	mound topography and less snow

TABLE 11. Changing the default settings for the EISMINT II experiments in PISM.

Option	**Default values [expers]**	**Units**	**Meaning**
`-Mmax`	0.5 [ABDEFGHIK], 0.25 [CJL]	m/a	max value of accumulation rate
`-Rel`	450 [ABEFGHIK], 425 [CDJL]	km	radial distance to equilibrium line
`-Sb`	10^{-2} [*all*]	(m/a)/km	radial gradient of accumulation rate
`-ST`	1.67×10^{-2} [*all*]	K/km	radial gradient of surface temperature
`-Tmin`	238.15 [ACDEGHIJKL], 243.15[B], 223.15[F]	K	max of surface temperature
`-track_Hmelt`			compute effective thickness of basal melt water (override default for EISMINT II)
`-Lz`	4500 [AE], 4000 [BCD], 5000 [F], 3000 [G]	m	height of the computational box

9. Realistic ice sheet and ice shelf modelling: two examples

9.1. **Modelling the Greenland ice sheet.** In this subsection we give an extended example of how to use PISM to model the Greenland ice sheet using somewhat out-of-date data. That is, we perform the ice sheet modelling experiment (intercomparison) known as EISMINT-Greenland [55]. As stated earlier, real data is not something we can freely distribute under the GNU Public License. The data for performing this intercomparison is, however, freely available at

`http://homepages.vub.ac.be/~phuybrec/eismint/greenland.html`

The data for this intercomparison is out of date, but this fact does not reduce its usefulness as an intercomparison or for a tutorial example in this manual.[11] The snow-fall accumulation map, ablation parameterization, surface temperature formula, surface elevation, and bedrock elevation maps are essentially as in the 1991 papers [38, 46]. In the ice and sea floor-core driven "forced climate" run `-ccl3` described below, the ice sheet is forced by changes in temperature from the GRIP core [15] and by changes in sea level from SPECMAP [32].

Substantial developments have occurred in modelling the Greenland ice sheet since the EISMINT-Greenland intercomparison. For example, a parameter-sensitivity study of a Greenland ice sheet model is described in [56]. The relation between Greenland ice sheet modelling, Earth deformation under ice sheet loads, and the reconstruction of global ice loading is analyzed in [65]. The response of Greenland ice sheet models to climate warming is described in [30] and [20].

Obtaining and converting EISMINT-Greenland data. This subsection describes the use of two Python scripts to convert the EISMINT-Greenland data into NetCDF files useable by PISM.

The Python scripts `eis_green.py` and `eis_core.py` can be found in the `pism/test` directory. In order to use the scripts, you must have downloaded the following ascii data files from the above web site:

- `grid20-EISMINT` (or `grid40-EISMINT`)
- `suaq20-EISMINT` (or `suaq40-EISMINT`)
- `specmap.017`
- `sum89-92-ss09-50yr.stp`

The Python libraries `numpy` (`http://numpy.scipy.org/`) and `pycdf` (`http://pysclint.sourceforge.net/pycdf/`) must also be present. Note that the scripts `eis_green.py` and `eis_core.py` can both take option `--prefix=foo/` to specify that the downloaded data is in directory `foo/`. The script `eis_green.py` has an option (`-g`) which allows the modeler to specify that either the 20 km (default) or 40 km data be converted to NetCDF; use `-g 40` for the coarser data.

Run

```
eis_green.py --prefix=foo/ -g 20
```

The NetCDF file `eis_green20.nc` (or `eis_green40.nc`) will be created, in the current directory, from the data in `grid20-EISMINT` and `suaq20-EISMINT`. It contains variables for the gridded latitude (`lat`), longitude (`lon`), surface altitude (`h`), bedrock altitude (`b`), and ice thickness (`H`). These values can be viewed graphically with `ncview` or as long text files with `ncdump`.

[11] Indeed the reader is encouraged to go and build a better Greenland model than the tutorial example here!

As an exercise, the NCO (`http://nco.sourceforge.net/`) can be used on `eis_green20.nc` to compare the putative ice surface elevation to the sum of the ice thickness and the bed elevation:

```
ncap -O -vs "check=h-(H+b)" eis_green20.nc eis_green_check.nc
```

The variable `check` in the output file holds the difference of `h` and `b + H`. Viewing this variable will show that `h` is within 1 meter of `b + H`. Thus the surface elevation `h` is consistent. It is also redundant: at bootstrapping PISM simply reads ice thickness `H` and bed elevation `b` and computes ice surface elevation as the sum of these two; the ice surface elevation variable `h` is not actually read from the bootstrapping NetCDF file.

It also should be noted that bed elevation (`b`) contains a missing value attribute of 0.0 because in several places (deep fjords, apparently) the bed elevation was not measured or trusted. When viewing this data in `ncview`, these values show as white spots. If these missing values are left in then the bed elevation map is extraordinarily rough and this makes reasonable ice flow predictions essentially impossible. We therefore suggest smoothing this aspect of the data. This can be done with another script named `fill_missing.py`, found in directory `pism/test/`. To use this script, the following command can be run:

```
fill_missing.py -i eis_green20.nc -v b -o eis_green_smoothedbed.nc
```

Here, `eis_green20.nc` is the input file, `b` is the variable with missing values, and `eis_green_smoothed.nc` is the output file. The script will look for an attribute named `missing_value` and fill in the missing values according to the averages of its neighbors. Multiple variables can be fixed at the same time by separating the variable names by commas. (That is, `fill_missing.py -i data.nc -v b,H,Ts -o data_smoothed.nc`. Note that *each* of the listed variables must have an attribute named `missing_value`.)

In addition to getting the EISMINT gridded data into a NetCDF format and filling missing values, there is an issue with Ellesmere Island. Ellesmere Island is very close to Greenland, and so it is possible for the modelled ice sheet to flow onto to it, and indeed this presumably occurred at the last glacial maximum. Since we don't, however, have correct topography or accumulation rates for Ellesmere Island we want to prevent this from happening. Therefore special code runs when `pgrn` is used (see below) which says that all points northwest of the line connecting the points $(68.18°E, 80.1°N)$ and $(62°E, 82.24°N)$ are removed from the flow simulation. The same applies to anything east of $30°E$ and south of $67°N$ so that the flow could not spread to the tip of Iceland (not likely). ("Removed from the flow simulation" actually means marked as ice-free ocean; note the use of option `-ocean_kill` below.)

Recall we mentioned the script `eis_core.py` as well. Just execute it, possibly giving the directory where the ice core and sea bed core data `specmap.017` and `sum89-92-ss09-50yr.stp` are found, if that is not the current directory:

```
eis_core.py --prefix=foo/
```

Two NetCDF files with one-dimensional (time series) data will be created, namely `grip_dT.nc` and `specmap_dSL.nc`. These can be `ncview`ed or `ncdump`ed.

Bootstrapping with EISMINT-Greenland data. Once the EISMINT Greenland data is obtained and converted to NetCDF, bootstrapping can begin. This 100 year run is a basic version, shown with `-verbose` to suggest the many aspects of "bootstrapping":

```
$ pgrn -bif eis_green_smoothedbed.nc -Mx 83 -My 141 -Mz 101 -y 10 -o grn10yr_20km -verbose
PGRN (EISMINT Greenland mode)
bootstrapping by PISM default method from file eis_green_smoothedbed.nc
  polar stereographic found: svlfp = -41.14, lopo =  71.65, sp =  71.00
  time t found in bootstrap file; using to set current year
  using default value Lz=4000.000000 for vertical extent of computational box for ice
  WARNING: ignoring values found for surface elevation h and using h = b + H
  WARNING: surface temperature Ts not found; using default  263.15 K
  WARNING: geothermal flux ghf not found; using default  0.042 W/m^2
  uplift not found. Filling with zero
  Hmelt not found. Filling with zero
  done reading .nc file
  determining mask by floating criterion; grounded ice marked as SIA (=1)
  setting accumulation in ice shelf-free ocean to default value -20.00 m/a
  filling in temperatures at depth using surface temperatures and guesses
bootstrapping done
geothermal flux vGhf is being set to: 0.050000
  [computational box for ice: ( 1640.00 km) x ( 2800.00 km) x ( 4000.00 m)]
  [grid cell dimensions     : (   20.00 km) x (   20.00 km) x (   40.00 m)]
  IceParam: Mx = 83, My = 141, Mz = 101, Mbz = 1,
            Lx = 820.00 km, Ly = 1400.00 m, Lz = 4000.00 m, Lbz =   0.00 m,
            dx = 20.000 km, dy = 20.000 km, dz = 40.000 m, year =   0.0000,
            history = ****************
bueler@ubuntu 2007-08-06 00:10:07 AKDT : pgrn -bif eis_green_smoothedbed.nc
-Mx 83 -My 141 -Mz 101 -y 10 -o grn10yr_20km -verbose
            *************************
            YEAR (+     STEP[N$]):     VOL    AREA    MELTF     THICK0     TEMP0
$$$$$      0.000 (+  0.00000[0 ]):   2.825   1.671    0.207   3042.000   273.173
    (volume of ice which is SIA, stream, shelf:     2.825,     0.000,     0.000)
  d(volume)/dt of ice (km^3/a):           0.00
  average value of dH/dt (m/a):        0.00000
  area percent covered by ice:         35.6917
    (area percent ice SIA, stream, shelf:         99.8564,    0.0000,    0.1436)
  max diffusivity D on SIA (m^2/s):      0.000
  max |bar U| in all ice (m/a):          0.000
    (av |bar U| in SIA, stream, shelf (m/a):        0.000,     0.000,     0.000)
  maximum |u|,|v|,|w| in ice (m/a):      <N/A>
  fraction of ice which is original:     0.989
$v$tf      0.128 (+  0.12831[0d]):   2.825   1.879    0.185   3041.841   270.516
    (volume of ice which is SIA, stream, shelf:     2.825,     0.000,     0.000)
  d(volume)/dt of ice (km^3/a):         419.30
  average value of dH/dt (m/a):        0.25096
  area percent covered by ice:         40.1350
    (area percent ice SIA, stream, shelf:         99.8723,    0.0000,    0.1277)
  max diffusivity D on SIA (m^2/s):     13.152
  max |bar U| in all ice (m/a):        739.232
```

```
    (av |bar U| in SIA, stream, shelf (m/a):       42.225,     0.000,     9.982)
  maximum |u|,|v|,|w| in ice (m/a):    712.745,   477.121,   137.953
  fraction of ice which is original:     0.965
...
$v$tf     10.000 (+  0.10510[0t]):   2.831   1.873    0.187   3033.913   270.523
    (volume of ice which is SIA, stream, shelf:     2.830,     0.000,     0.000)
  d(volume)/dt of ice (km^3/a):         578.85
  average value of dH/dt (m/a):        0.30902
  area percent covered by ice:         40.0154
    (area percent ice SIA, stream, shelf:         99.9146,    0.0000,    0.0854)
  max diffusivity D on SIA (m^2/s):      5.054
  max |bar U| in all ice (m/a):        370.554
    (av |bar U| in SIA, stream, shelf (m/a):       33.263,     0.000,    14.683)
  maximum |u|,|v|,|w| in ice (m/a):    331.165,   291.717,     8.628
  fraction of ice which is original:     0.964
done with run ...
Writing model state to file 'grn10yr_20km.nc' ... done.
```

Since the EISMINT Greenland data does not contain certain variables necessary to initialize PISM in the sense of initial values for the relevant partial differential equations, this bootstrapping mode fills in several default values. (As noted in section 5 we are doing naive inverse modeling.) For instance, the variables `Ts`, `ghf`, `uplift`, and `Hmelt` were not found in the `-bif` file. Thus, these variables were filled with default `263.15`, `0.042`, `0`, and `0` respectively. Also, recall that the EISMINT Greenland data had redundant surface elevation (`h`) values, we see that `h` is ignored and PISM uses the sum of bed elevation and ice thickness "`h = b + H`".

Running the EISMINT-Greenland steady state experiments. There are several experiments that can be run using command line arguments with `pgrn`. Some of them require the use of more python scripts, while single line commands are sufficient for others.

The first experiment is a steady state run "SSL2" which uses the parameters specified in the EISMINT-Greenland description [55]. This experiment is intended to be run until the model reaches a "steady state", which means a particular standard for a small volume change rate (less than a .01% change in volume in 10,000 years). To run this experiment, there is a script that automates the process so that it is not necessary to manually check the changes in volume . This script is called `ssl_exper.py` and can be found in the `pism/test` directory. When run, it will continue, saving states every 10,000 years, until the change in volume is less than .01% between 10,000 year runs. It can be used with multiple processors by setting the option `-n`, for example:

```
ssl_exper.py -n 8
```

One can also give a `-ssl3 -gk` option to `pgrn`, or a `--ssl3` option to `ssl_exper.py`. This gives a particular interpretation to the SSL3 experiment which is described in [55], namely that the Goldsby-Kohlstedt flow law [18] with an enhancement factor of 1.0 will be used. (There is no assertion that this choice is superior at this point.)

Climate forcing from GRIP and SPECMAP. The next experiment is intended to be carried out after the completion of the steady state run (above) using the saved steady model state.

Recall that the NetCDF files `grip_dT.nc` and `specmap_dSL.nc` contain time series data for change in surface temperature and the change in sea level. The options `-dTforcing` and `-dSLforcing` for `pgrn` can use these data for a "CCL3" run [55] under PISM. Before every time step, `pgrn` adds in a change to the surface temperature and sea level in order to incorporate the effects of global climate changes. The data in `grip_dT.nc` goes about 250,000 years into the past, while the data in `specmap_dSL.nc` goes back about 780,000 years, but EISMINT-Greenland specifies that the run will start at the beginning of the GRIP data. Thus we manually set the starting year using the option `-ys`. The CCL3 experiment can be run using the following command:

```
pgrn -ccl3 -if green20_SSL3.nc -dTforcing grid_dT.nc \
  -dSLforcing specmap_dSL.nc -ys -249900 -ye 0 -o green20_CCL3_y0
```

EISMINT-Greenland also calls for a baseline run for another 500 years which starts from the end of the CCL3 run and has steady current climate forcing. If `pgrn -ccl3` is run past year 0, and thus past then end of the GRIP and SPECMAP data, this baseline run will occur by default. So the previous command can be run for another 500 years, and the forcing can be omitted:

```
pgrn -ccl3 -if green20_CCL3_y0.nc -y 500 -o green20_CCL3_y500
```

A greenhouse warming scenario. A final "greenhouse warming" experiment "GWL3" is described in the EISMINT-Greenland [55]. It uses the model state obtained for present day from the CCL3 run. Using the previous commands, this is the file named `green20_CCL3_y0.nc`. The GWL3 experiment (option `-gwl3`) is intended to be run for 500 years with the temperature increasing by $0.035^\circ C$/year for the first 80 years, then at a rate of $0.0017^\circ C$/year for the last 420 years:

```
pgrn -gwl3 -if green20_CCL3_y0.nc -y 500 -o green20_GWL3
```

9.2. **Validation of PISM as a flow model for the Ross ice shelf.** Recall that we use the term "validation" to describe the comparison of model output with physical observations in cases where those physical observations are believed to be sufficiently complete and of sufficient quality so that the performance of the numerical model can be assessed. Roughly speaking, in validation the observations or data are better than the model, so the comparison tells one about the quality of the model and not the quality or incompleteness of the data.

As part of the first EISMINT series of intercomparisons, MacAyeal and others [41] performed a validation of several ice shelf numerical models using the Ross ice shelf as an example. We will refer to this intercomparison and its associate write-up [41] as "EISMINT-Ross". The models were compared to data from the RIGGS (= Ross Ice shelf Geophysical and Glaciological Survey) [5, 4]. The RIGGS data were acquired in the period 1973–1978. In particular, the RIGGS data include the (horizontal) velocity of the ice shelf measured at a few hundred locations in a reasonably regular grid across the shelf; see figure 5 below for an indicaiton of these positions.

Substantial developments have occurred in the modelling of the Ross ice shelf since the EISMINT-Ross intercomparison. For example, inverse modelling techniques were used to recover

depth-averaged viscosity of the Ross ice shelf from the RIGGS data [59] and a parameter-sensitivity study was performed for a particular Ross ice shelf numerical model [28].

Bringing in the data. Using these data to validate PISM's ice shelf modelling abilities requires downloading (text) data files from the website `http://homepages.vub.ac.be/~phuybrec/eismint/iceshelf.html`. In particular, these files must be downloaded:

- `111by147Grid.dat`
- `kbc.dat`
- `inlets.dat`

In the next step we will assume that these three files are in a directory `eisDownload/`, and we will refer to these as the "EISMINT-Ross" data. Note that these data are for a particular rectangular grid with 6.822km spacing in both x and y directions, but that (as usual) the PISM computational grid is determined by the user.

The first step for using these data, as usual for PISM, is to convert them to a machine readable NetCDF file. There is already a script `eis_ross.py` for this purpose in subdirectory `pism/test/ross/`. It reads the above three EISMINT-Ross `.dat` files and it creates a NetCDF file with the data and some metadate needed by PISM. It is used this way:

```
test/ross/eis_ross.py --prefix=eisDownload/ -o ross.nc
```

The output file can be reconverted to text (CDL) using `ncdump` or it can be viewed graphically with `ncview`.

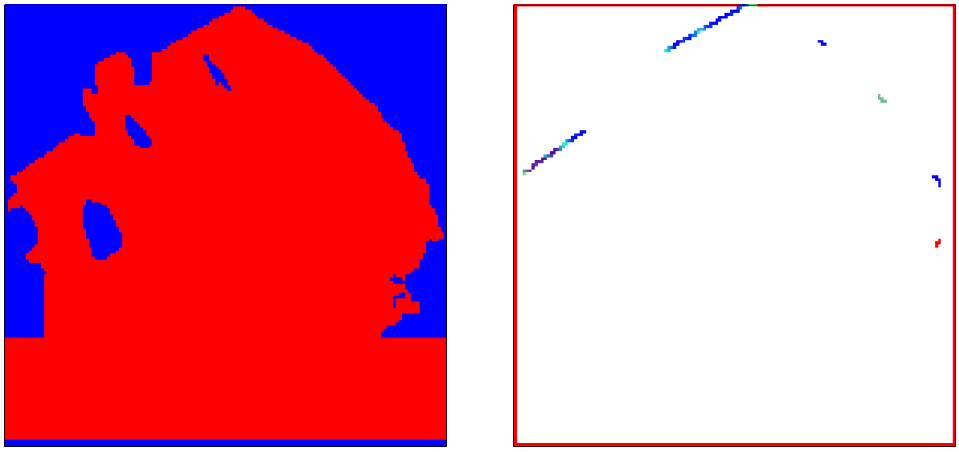


FIGURE 3. Two views from `ncview` of the EISMINT-Ross data in the NetCDF file `ross.nc`. The floating-versus-grounded mask (left; red areas are floating ice shelf) and the x-component of the non-zero kinematic (Dirichlet) boundary condition for velocity (right).

The NetCDF file `ross.nc` contains ice thickness, bed elevations, surface temperature, and accumulation, and all of these are typical of ice sheet modelling data. It also has variables `ubar` and `vbar` which give the boundary values which are needed for the diagnostic computation of velocity. Also present are the variables `mask`, which shows the domain where the ice shelf is modelled, and `bcflag`, which shows the locations where the boundary conditions are to

be applied. Note that the original EISMINT-Ross data are on a 111 by 147 grid but that `eis_ross.py` extends this grid to a more convenient 147 by 147 grid, with the same 6.822km spacing, which has ice-free ocean beyond the (straightened; compare [41]) calving front. See figure 3.

Diagnostic computation of ice shelf velocity. The basic Ross ice shelf velocity computation from these data might be:

```
pismd -ross -bif ross.nc -ssaBC ross.nc -Mx 147 -My 147 -Mz 11 -mv
```

Here we bootstrap (`-bif`) from `ross.nc`. We also use option `-ssaBC` to specify `ross.nc` as the source of the boundary value data for the ice shelf equations, as mentioned in the last paragraph. The computational grid specified here is the 6.822 km data grid in EISMINT-Ross with 147 grid points in each direction. Note that the small number of vertical levels (`-Mz 11`) is reasonable because the EISMINT-Ross intercomparison specifies that the temperature at depth be the surface temperature [41], and thus there is no thermocoupling issue.

At the end of this run the computed velocity field is compared to the interpolated data on observed velocities in the file `ross.nc`. (In particular, that NetCDF file contains a variable `accur` which identifies the region in which the observed velocities are valid. The observed (interpolated) velocities themselves are stored in variables `mag_obs` and `azi_obs`.) We see that the largest computed velocity in the ice shelf is just under a thousand meters per year, and that errors in the vector values of velocity are roughly 10% in a sum of squares average sense. (There is more on viewing the results, on measuring the correspondence with measurements, and on tuning the hardness parameter to fit the data, below.)

There are many variations on this basic "`pismd -ross`" run above. First of all one can get more information during the run by adding diagnositic viewers and a more complete (verbose) report to standard out:

```
pismd -ross -bif ross.nc -ssaBC ross.nc -Mx 147 -My 147 -Mz 11 -mv \
  -d cnmu -verbose 4 -pause 10
```

Secondly one might want to do the run in parallel, do it on a finer grid, and ask for higher tolerance. For example,

```
mpiexec -n 6 pismd -ross -bif ross.nc -ssaBC ross.nc -Mx 201 -My 201 -Mz 11 -mv
```

The result is not exactly the same [but really good; FIXME].

Alternately one might want to change the hardness parameter from the default value $\bar{B} = 1.9 \times 10^8\,\mathrm{Pa\,s}^{1/3}$ [41] to a slightly softer value because the highest computed velocity with the default setting is a little lower than the maximum measured in the RIGGS data (namely, 1007 m/a; see files `pism/test/ross/README` and `pism/test/ross/riggs_clean.dat`). We can also use lower (more severe) tolerances for the nonlinear iteration (`-mv_rtol`) and the linear iteration (`-ksp_rtol`) to get more confidence in the numerical scheme. Finally we can also save the model state with a meaningful name:

```
pismd -ross -bif ross.nc -ssaBC ross.nc -Mx 147 -My 147 -Mz 11 -mv \
  -verbose 4 -constant_hardness 1.8e8 -mv_rtol 1e-6 -ksp_rtol 1e-10 \
  -o ross_out_1p8
```

Using `ncview` on `ross_out_1p8.nc` to view the horizontal speed of the ice shelf gives figure 4.

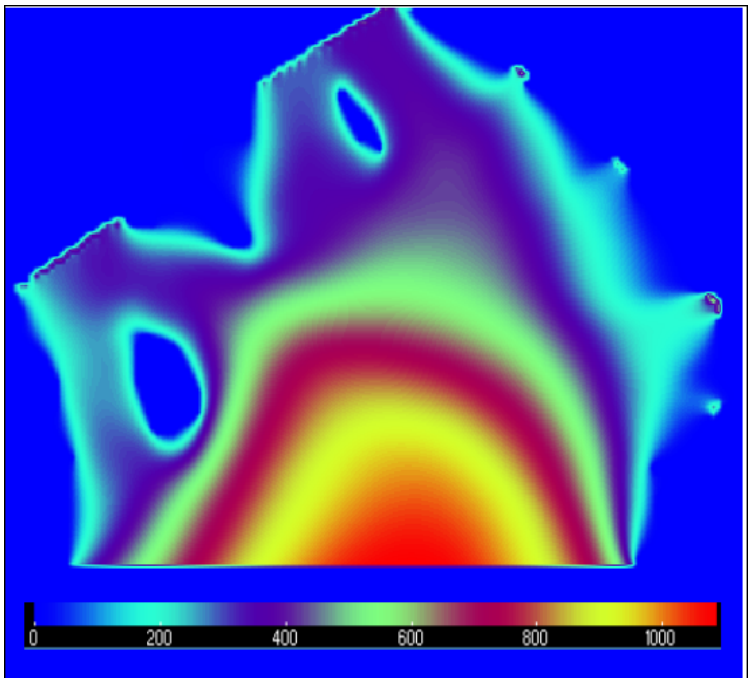


FIGURE 4. Computed horizontal speed of the Ross ice shelf using $\bar{B} = 1.8 \times 10^8\,\mathrm{Pa\,s}^{1/3}$. Color gives velocity in m/a; see scale.

TABLE 12. Non-obvious options available and/or recommended with `pismd -ross`.

Option	**Explanation/Comments**
`-d cnmu`	a good way to see what is going on
`-o foo -of nm`	writes NetCDF file `foo.nc` with full three-dimensional velocity fields *and* writes some model results to Matlab-readable file `foo.m`
`-pause N`	pause for N seconds when refreshing viewers
`-ross`	only use with executable `pismd`
`-tune x,y,z`	run through constant hardness values $\bar{B} = x, x+y, x+2y, \ldots, z(= x+ky)$; (note no spaces in "`x,y,z`")
`-verbose 4`	shows information on nonlinear iteration and Krylov solve and parameters related to solving the ice shelf equations

Table 12 lists some of the options which are useful for this kind of diagnositic velocity computation.

Comparison to RIGGS data. The file `riggs_clean.dat` in directory `pism/test/ross/` is a cleaned-up version of the original RIGGS data [5, 4]. To convert this data to a NetCDF file, as needed next, do

```
cd test/ross/
./eis_riggs.py -o riggs.nc
```

A file `riggs.nc` will be created. Note that the data is one-dimensional in this NetCDF file, that is, it is just some lists of values with an index dimension `count`. (See `pism/test/ross/README` for more explanation on this RIGGS data.)

Now, `pismd -ross` can read this data and compute a χ^2 statistic comparing model output to the data. Assuming one is in the `pism/test/ross/` directory, and that both `ross.nc` and `riggs.nc` are in this directory, this command will produce a χ^2 statistic:

```
$ pismd -ross -bif ross.nc -ssaBC ross.nc -Mx 147 -My 147 -Mz 11 -mv -riggs riggs.nc
PISMD: initializing EISMINT Ross ice shelf velocity computation ...
...
maximum computed speed in ice shelf is    951.113 (m/a)
...
comparing to RIGGS is data in riggs.nc ...
Chi^2 statistic for computed results compared to RIGGS is   5390.542
 ... done.
```

Naturally, the question is "does this χ^2 value of 5390.5 represent a good fit of model result to observations"? Also naturally, there is no objective answer. For comparison, Table 1 in [41] is reproduced here as Table 13. As noted, all these results are with a constant hardness parameter $\bar{B} = 1.9 \times 10^8\,\mathrm{Pa\,s^{1/3}}$ [41]. The maximum computed horizontal ice speed above of 951.1 m/a is lower than the maximum velocities reported by the other models but, on the other hand, the maximum measured speed in the RIGGS data set is 1007 m/a (near the calving front, of course).

TABLE 13. Model performance index, χ^2 (non-dimensional). *(Reproduction of Table 1 in* [41]*.)*

Model	χ^2	*Maximum velocity* $\mathrm{m\,a^{-1}}$
Bremerhaven1	3605	1379
Bremerhaven2	12 518	1663
Chicago1	5114	1497
Chicago2	5125	1497
Grenoble	5237	1508

Tuning the ice hardness for a better fit to RIGGS. Because there is a relatively rich data set from RIGGS on ice velocity, it is reasonable to ask whether the PISM computed velocities can fit the data better if the hardness parameter $\bar{B}$ is adjusted. There is a Python script `pism/test/ross/tune.py` which runs `pismd -ross` with several values of $\bar{B}$. By default it uses smaller values for the convergence tolerances, and it can be run with multiple processors. We see that with a hardness $\bar{B} = 1.7 \times 10^8\,\mathrm{Pa\,s^{1/3}}$ we get a maximum computed speed of 1256.8 m/a and a χ^2 of 3398.0. To the extent that we are comparing apples to apples, so to speak, this is more-or-less as good as any of the results reported [41].

```
$ ./tune.py -n 2
TUNE.PY (for EISMINT Ross; compare to table 1 in MacAyeal et al 1996)
```

```
trying "mpiexec -n 2 pismd -ross -bif ross.nc -ssaBC ross.nc -riggs riggs.nc
-ksp_rtol 1e-08 -mv_rtol 1e-05 -Mx 147 -My 147 -Mz 11 -mv -constant_hardness 1.5e8"
  finished in 108.9859 seconds; max computed speed and Chi^2 as follows:
  |maximum computed speed in ice shelf is   1749.024 (m/a)
  |Chi^2 statistic for computed results compared to RIGGS is  10153.2
trying "mpiexec -n 2 pismd -ross -bif ross.nc -ssaBC ross.nc -riggs riggs.nc
-ksp_rtol 1e-08 -mv_rtol 1e-05 -Mx 147 -My 147 -Mz 11 -mv -constant_hardness 1.6e8"
  finished in 107.7356 seconds; max computed speed and Chi^2 as follows:
  |maximum computed speed in ice shelf is   1471.820 (m/a)
  |Chi^2 statistic for computed results compared to RIGGS is   4833.6
trying "mpiexec -n 2 pismd -ross -bif ross.nc -ssaBC ross.nc -riggs riggs.nc
-ksp_rtol 1e-08 -mv_rtol 1e-05 -Mx 147 -My 147 -Mz 11 -mv -constant_hardness 1.7e8"
  finished in 104.5976 seconds; max computed speed and Chi^2 as follows:
  |maximum computed speed in ice shelf is   1256.762 (m/a)
  |Chi^2 statistic for computed results compared to RIGGS is   3398.0
trying "mpiexec -n 2 pismd -ross -bif ross.nc -ssaBC ross.nc -riggs riggs.nc
-ksp_rtol 1e-08 -mv_rtol 1e-05 -Mx 147 -My 147 -Mz 11 -mv -constant_hardness 1.8e8"
  finished in 105.8533 seconds; max computed speed and Chi^2 as follows:
  |maximum computed speed in ice shelf is   1087.281 (m/a)
  |Chi^2 statistic for computed results compared to RIGGS is   3917.0
trying "mpiexec -n 2 pismd -ross -bif ross.nc -ssaBC ross.nc -riggs riggs.nc
-ksp_rtol 1e-08 -mv_rtol 1e-05 -Mx 147 -My 147 -Mz 11 -mv -constant_hardness 1.9e8"
  finished in 100.2162 seconds; max computed speed and Chi^2 as follows:
  |maximum computed speed in ice shelf is    951.721 (m/a)
  |Chi^2 statistic for computed results compared to RIGGS is   5382.4
trying "mpiexec -n 2 pismd -ross -bif ross.nc -ssaBC ross.nc -riggs riggs.nc
-ksp_rtol 1e-08 -mv_rtol 1e-05 -Mx 147 -My 147 -Mz 11 -mv -constant_hardness 2.0e8"
  finished in 102.3863 seconds; max computed speed and Chi^2 as follows:
  |maximum computed speed in ice shelf is    841.752 (m/a)
  |Chi^2 statistic for computed results compared to RIGGS is   7266.9
```

Additional visualization using MATLAB *output from PISM.* The visualization abilities of PISM's runtime viewers and of `ncview` (applied to the PISM output NetCDF file) are limited. PISM can save certain variables in a MATLAB-readable form, however, and this is a situation where it might be useful.

```
pismd -ross -bif ross.nc -ssaBC ross.nc -riggs riggs.nc \
  -ksp_rtol 1e-08 -mv_rtol 1e-05 -Mx 147 -My 147 -Mz 11 -mv \
  -constant_hardness 1.7e8 -o ross1p7 -of m
```

When this completes, make sure MATLAB can find `ross1p7.m` and also files `ross_plot.m` and `riggs_clean.dat` (both in directory `pism/test/ross/`). Execute the `m`-files as commands:

```
>> ross1p7
>> ross_plot
```

Figure 5 is the result. We have succeeded in modeling a real ice shelf.

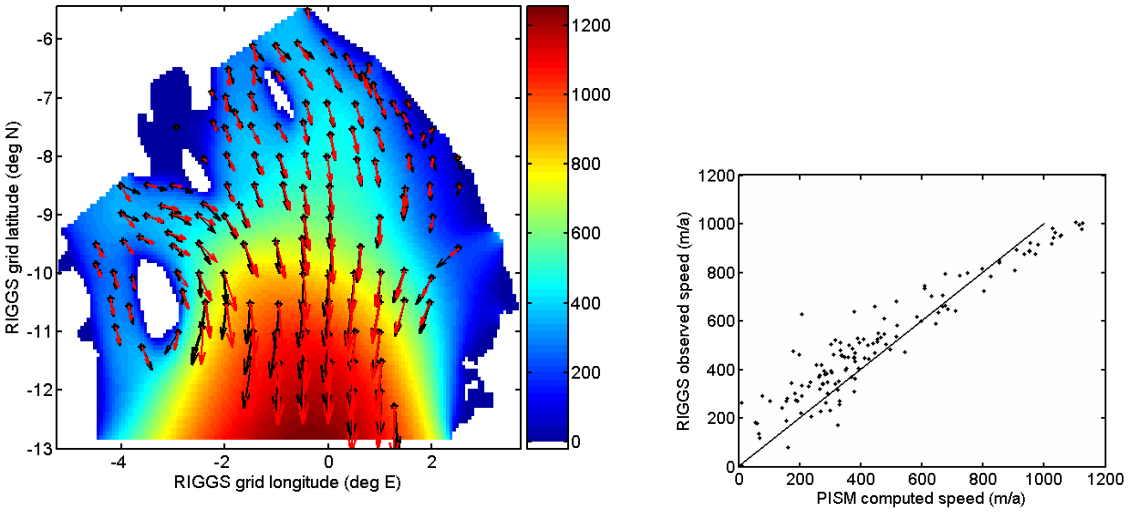


FIGURE 5. *Left*: Color is speed in m/a. Arrows are observed (black) and computed (red) velocities at RIGGS points. *Right*: Comparison between modelled and observed speeds at RIGGS points; compare figure 2 in [41].

10. Inside PISM: overviews of models, schemes, and sources

10.1. **The continuum models in PISM.** Significant features of the continuum model approximated by the PISM include:

- The inland ice sheet is modeled with the thermocoupled shallow ice approximation equations [17], and some temperature-activated basal sliding is allowed.
- Ice shelves and ice streams are modeled by shallow equations which describe flow by longitudinal strain rates and basal sliding. These equations are different from the shallow ice approximation. In shelves and streams the velocities are independent of depth within the ice. The equations were originally established for ice shelves [43, 44, 41]. They were adapted for ice streams, as "dragging ice shelves," by MacAyeal [40]; see also [27].
- The regions of grounded ice in which the ice stream model is applied can be determined from mass balance velocities [3], from observed surface velocities, or from a plastic till assumption and the associated free boundary problem [62].
- A three dimensional age field is computed.
- A temperature model for the bedrock under an ice sheet is included.
- Geothermal flux which varies in the map-plane can be used, for instance based on the new Antarctic results in [64] and [42].
- Within the shallow ice sheet regions the model can use the constitutive relation of Goldsby and Kohlstedt [18, 53]. For inclusion in this flow law, grain size can be computed using a age-grain size relation from the Vostok core data [16], for example.
- The Lingle-Clark [11, 39] bed deformation model can be used. It combines a spherical elastic earth and viscous half-space asthenosphere/mantle. It generalizes the better known elastic lithosphere, relaxing asthenosphere ("ELRA") model [21]. This model can be initialized by an observed bed uplift map [11], or even an uplift map computed by an external model like that in [33].

Many of the parts of the model described above are optional. For instance, the Paterson-Budd-Glen [48] flow can replace the Goldsby-Kohlstedt law, a simple isostasy model can be substituted for the more sophisticated one, and so on. Options can be chosen at the command line as described in section A above.

The following features are *not* included in the continuum model, and would (or will) require major additions:

- Inclusion of all components of the stress tensor (i.e. longitudinal stresses within the shallow ice approximation region and additional shear stress components in shelves/streams) through either an intermediate order scheme [6, 25, 60] or the full Stokes equations [17].
- A model for water–content within the ice. In the current model the ice is *cold* and not *polythermal*; compare [19]. On the other hand, in the current model the energy used to melt the ice within a given column, if any, is conserved. In particular, a layer of basal melt water evolves by conservation of energy in the column. This layer can activate basal sliding and its latent heat energy is available for refreezing.
- A model for basal water mass conservation in the map-plane; compare [35].

- A fully spherical Earth deformation model, for example one descended from the Earth model of [52], like the one used to estimate current Antarctic uplift in [33].

10.2. **The numerical schemes in PISM.** Significant features of the numerical methods in PISM include:

- Verification [58] is a primary concern and is built into the code. Nontrivial verifications are available for isothermal ice sheet flow [12], thermocoupled sheet flow [9, 10], the earth deformation model [11], the coupled (ice flow)/(earth deformation) system in an isothermal and pointwise isostasy case [11], and the MacAyeal equations for ice stream flow [62, 7].
- The code is *structurally* parallel. In fact, the PETSc toolkit is used at all levels [1]. PETSc manages the MPI-based communication between processors, and provides an interface to parallel numerical linear algebra and other numerical functions.
- The grid can be chosen at the command line. Regridding can be done at any time, for example taking the result of a rough grid computation and interpolating it onto a finer grid.
- A moving boundary technique is used for the temperature equation which does not stretch the vertical in a singular manner; the Jenssen [34] change of variables is not used.
- The model uses an explicit time stepping method for flow and a partly implicit method for temperature. Advection of temperature is upwinded [45]. As described in [10], reasonably rigorous stability criteria are applied to the time-stepping scheme, including a diffusivity-based criteria for the explicit mass continuity scheme and a CFL criteria [45] for temperature/age advection.
- The local truncation error is generically first order (i.e. $O(\Delta x, \Delta y, \Delta z, \Delta t)$).
- The shallow shelf approximation (SSA) [66] is used to determine velocity in the ice shelf and ice stream regions. Like the full non-Newtonian Stokes equations, they are nonlinear and nonlocal equations for the velocity given the geometry of the streams and shelves and given the ice temperature. They are solved by straightforward iteration of linearized equations with numerically-determined (vertically-averaged) viscosity. Either plastic till or linear drag is allowed. As with all of PISM, the numerical approximation is finite difference. The linearized finite difference equations are solved by any of the Krylov subspace methods in PETSc [7, 1].
- The bed deformation model is implemented by a new Fourier collocation (spectral) method [11].
- Implementation is in C++ and is object-oriented. For example, verification occurs in a derived class.

10.3. **The PISM source code.** This ice sheet model is implemented as a collection of C++ object classes, the most central of which is `IceModel`. Actually there are three basic classes to know about, all of which are in `pism/src/base/`:

- `IceGrid`, which describes the shape of the grid and parallel layout. This abstraction could be used to streamline transferring model data between different grids.

- `MaterialType`. Various materials are derived from the class `MaterialType` which merely defines a couple physical constants. `IceType` is a derived class, but still an abstract class. which defines the physical properties of ice as a material. Concrete classes derived from `IceType` are `ThermoGlenIce` (which uses the EISMINT constants) and `GKIce` as well as `HybridIce`, which implements the Goldsby-Kohlstedt flow law [18].[12] Similarly, there is `BedrockType` and `OceanType` which merely define associated physical constants.
- `IceModel` which contains myriad data structures, flags, PETSc `Vec`s (below), and most of the methods and numerical methods of PISM.

The methods for `IceModel` are many, as noted. In addition to numerics, they initialize the model (from input data files or from formulas describing various exact solutions), they read user options, they allocate arrays in a distributed manner under PETSc (below), they control diagnostic viewers, they run the adaptive time-stepping mechanism, they report the state of PISM to the user, and they write out the model state to files.

A derived class of `IceModel` called `IceCompModel` is used for verification. It has additional structures which allows `IceModel` to have compensatory sources and compute initial conditions from, and especially to report errors relative to, known exact solutions [12, 10, 9]. Another derived class used for verification is `IceExactStreamModel`, which implements the exact solution described in [62].

Other derived classes of `IceModel` are `IceEISModel`, `IceGRNModel`, and `IceROSSModel`. These correspond to simplified geometry experiments, the EISMINT-Greenland, and the EISMINT-Ross ice shelf models described in this manual.

There are five established drivers which in essense just call constructors and destructors for `IceModel` or a derived class thereof, and tell it to go. These are in the source files `pgrn.cc`, `pismd.cc`, `pismr.cc`, `pisms.cc`, and `pismv.cc`. These correspond to the executables `pgrn`, `pismd`, `pismr`, `pisms`, and `pismv`, respectively. These drivers do no computation but differ in which derived class is constructed and in how certain options are handled. The driver `pismr.cc` and its executable `pismr` only use the base class `IceModel`; it requires a pre-existing saved PISM model state to start.

The different derived classes generally invoke the same numerical procedures to handle the various partial differential equations of the continuum model. In the case of `IceCompModel`, this fact is at the heart of the verification mechanism.

10.4. **PETSc: An overview for PISM users.** The PETSc library [1, 2] provides essential support for distributed arrays and linear solvers in a parallel computing environment. "PETSc" stands for "Portable, Extensible Toolkit for Scientific Computation." It is a suite of data structures and routines in C for the scalable parallel solution of partial differential equations and their scientific applications. Large parts of PETSc relate especially to finite-difference-type regular, rectangular grids but PETSc has been used for unstructured grids as well. Documentation for PETSc is available from the web site at `http://www-unix.mcs.anl.gov/petsc/petsc-as/`.

[12] Actually, it is difficult or impossible to implement a complete Goldsby-Kohlstedt `IceType` because the inverse constitutive relation is required for computation of SSA velocity fields. `HybridIce` is Goldsby-Kohlstedt ice in the interior of the ice sheet and Glen ice in ice streams and shelves.

PETSc employs the MPI standard for all message-passing communication. PETSc is deliberately at a higher level of abstraction than MPI; PETSc protects the programmer from explicit consideration of message passing.

Most variables in a PETSc program are of newly-defined distributed types including

```
DA   da;
Vec  v;
KSP  ksp;
```

In fact most of the PETSc types merely declare pointers but they should be regarded as objects (abstract data types). The objects must be created with calls to functions like `DACreate2d()`, `VecCreate()`, etc. They should be destroyed when they are not needed with calls to corresponding `Destroy()` functions.

Distributed arrays and vectors. PETSc has an abstract date type called a Distributed Array (DA). Objects of type DA contain information about the grid and stencil. They can have information about coordinates, but the code does not use this feature at present. Vectors are created with `DAVecCreate()` and similar. These vectors will be distributed across the processors as indicated by the distributed array.

There are two parameters of note: stencil type and stencil width. The stencil types are `DA_STENCIL_STAR` and `DA_STENCIL_BOX`. They are generalizations of the five point and nine point stencils typical of two dimensional discretizations respectively. In particular, `DA_STENCIL_STAR` indicates that ghosted points (information owned by a different processor) will be needed only along the coordinate axes while `DA_STENCIL_BOX` indicates that ghosted points will be needed in the box shaped region surrounding each point. The stencil width indicates how many points in each direction will be needed. We never need a stencil width greater than 1 and only need BOX style stencils when gradient terms must be evaluated on a staggered grid (h in SIA velocity and $\bar{u}, \bar{v}$ in computation of effective viscosity in SSA velocity). Keeping all other two dimensional vectors on a STAR type stencil would reduce the necessary communication slightly, but would complicate the code. For this reason, all two dimensional vectors are kept on a box type distributed array.

The three dimensional distributed arrays are aligned so that they have the same horizontal extent as the associated two dimensional distributed array, but have complete vertical extent. One point of confusion is the redefinition of the x, y, z axes. Contrary to the PETSc default, our z axis changes most rapidly through memory while the x axis changes most slowly. That is, our C style arrays will be addressed as `u[i][j][k]` where `i,j,k` are the coordinate indices in the directions x, y, z respectively.

DA-based vectors can be accessed by `DAVecGetArray()` and restored with `DAVecRestoreArray()`. The resulting pointer should be addressed using normal multidimensional array indexing where values range over the global array.

PETSc DA based vectors can be "local" or "global". Local vectors include space for the ghosted points. That is, when `DAVecGetArray()` is called, the resulting array can be indexed on all the ghosted points. However, all vector operations act only on the local portion. `DALocalToLocalBegin()` and then `DALocalToLocalEnd()` should be called to update the ghost

points before they will be needed. Global vectors do not hold ghosted values, but array operations will act on the entire vector. Hence local vectors typically need to be mapped to global vectors before viewing or using in a linear system. This is achieved with `DALocalToGlobal()`.

Solving linear systems. PETSc is designed for solving large, sparse systems in a distributed environment. Iterative methods are the name of the game and especially Krylov subspace methods such as conjugate gradients and GMRES. For consistency, all methods use the Krylov subspace interface. For this, the user declares an object of type `KSP`. Various options can be set and the preconditioner context can be extracted. PETSc has an options database which holds command line options. To allow these options to influence the `KSP`, one should call `KSPSetFromOptions()` prior to solving the system. The default method is GMRES(30) with ILU preconditioning.

To solve the system, a matrix must be attached to the `KSP`. The first time `KSPSolve()` is called, the matrix will be factored by the preconditioner and reused when the system is called for additional right hand sides. The default matrix format is similar to the Matlab `sparse` format. Each processor owns a range of rows. Elements in matrices and vectors can be set using `MatSetValues()` and `VecSetValues()`. These routines use the global indexing and can set values on any processor. The values are cached until one calls `MatAssemblyBegin()` followed by `MatAssemblyEnd()` to communicate the values.

In the SSA velocity computation, the solution and right hand side vectors are not DA based.

The vector (field) components are interlaced and distributed. This seemed to be the most straightforward method to solve the system (as opposed to using more advanced features intended for multiple degrees of freedom on DA based vectors). This also allows the matrix to have an optimal parallel layout.

PETSc utility functions. The `PetscViewer` interface allows PETSc objects to be displayed. This can be in binary to disk, in plain text to the terminal, in graphical form to an X server, to a running instance of Matlab, etc. Typically, one will want to view an entire vector, not just the local portion, so DA based local vectors are mapped to global vectors before viewing.

When viewing multiprocessor jobs, the display may have to be set on the command line, for instance as `-display :0` or similar; this must be given as the final option. For example,

```
mpiexec -n 2 pismv -test C -Mx 101 -My 101 -Mz 31 -y 1000 -d HT -display :0
```

views a two processor run of test C.

PETSc allows the programmer to access command line arguments at any time during program execution. This is preferable to using `getopt.h` for this purpose.

Quite ellaborate error tracing and performance monitoring is possible with PETSc. All functions return `PetscErrorCode` which should be checked by the macro `CHKERRQ()`. Normally, runtime errors print traceback information when the program exits. If this information is not present, you may need to use a debugger which is accessible with the command line options `-start_in_debugger` and `-on_error_attach_debugger`. Also consider options such as `-log_summary` to get diagnostics written to the terminal.

References

[1] S. Balay and eight others, *PETSc users manual*, Tech. Rep. ANL-95/11 - Revision 2.3.2, Argonne National Laboratory, 2006.

[2] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, *Efficient management of parallelism in object oriented numerical software libraries*, in Modern Software Tools in Scientific Computing, E. Arge, A. M. Bruaset, and H. P. Langtangen, eds., Birkhäuser Press, 1997, pp. 163–202.

[3] J. L. Bamber, D. G. Vaughan, and I. Joughin, *Widespread complex flow in the interior of the Antarctic ice sheet*, Science, 287 (2000), pp. 1248–1250.

[4] C. R. Bentley, *Glaciological studies on the Ross Ice Shelf, Antarctica, 1973–1978*, Antarctic Research Series, 42 (1984), pp. 21–53.

[5] ———, *The Ross Ice shelf Geophysical and Glaciological Survey (RIGGS): Introduction and summary of measurments performed*, Antarctic Research Series, 42 (1984), pp. 1–20.

[6] H. Blatter, *Velocity and stress fields in grounded glaciers: a simple algorithm for including deviatoric stress gradients*, J. Glaciol., 41 (1995), pp. 333–344.

[7] J. Brown, *Verifying PISM*. UAF Master's project presentation `www.dms.uaf.edu/~bueler/slides_JedBrown.pdf`, 2006.

[8] E. Bueler, *Equilibrium ice sheets solve variational inequalities*. 2006.

[9] E. Bueler and J. Brown, *On exact solutions and numerics for cold, shallow, and thermocoupled ice sheets*. preprint `arXiv:physics/0610106`, 2006.

[10] E. Bueler, J. Brown, and C. Lingle, *Exact solutions to the thermomechanically coupled shallow ice approximation: effective tools for verification*, J. Glaciol., (2007). to appear.

[11] E. Bueler, C. S. Lingle, and J. A. Kallen-Brown, *Fast computation of a viscoelastic deformable Earth model for ice sheet simulation*, Ann. Glaciol., 46 (2007), pp. 97–105.

[12] E. Bueler, C. S. Lingle, J. A. Kallen-Brown, D. N. Covey, and L. N. Bowman, *Exact solutions and numerical verification for isothermal ice sheets*, J. Glaciol., 51 (2005), pp. 291–306.

[13] R. Calov and R. Greve, *Correspondence: A semi-analytical solution for the positive degree-day model with stochastic temperature variatons*, J. Glaciol, 51 (2005), pp. 173–175.

[14] N. Calvo et al., *On a doubly nonlinear parabolic obstacle problem modelling ice sheet dynamics*, SIAM J. Appl. Math., 63 (2002a), pp. 683–707.

[15] W. Dansgaard and ten others, *Evidence for general instability of past climate from a 250-kyr ice-core record*, Nature, 364 (1993), pp. 218–220.

[16] EPICA community members, *Eight glacial cycles from an Antarctic ice core*, Nature, 429 (2004), pp. 623–628. doi: 10.1038/nature02599.

[17] A. C. Fowler, *Mathematical Models in the Applied Sciences*, Cambridge Univ. Press, 1997.

[18] D. L. Goldsby and D. L. Kohlstedt, *Superplastic deformation of ice: experimental observations*, J. Geophys. Res., 106 (2001), pp. 11017–11030.

[19] R. Greve, *A continuum–mechanical formulation for shallow polythermal ice sheets*, Phil. Trans. Royal Soc. London A, 355 (1997), pp. 921–974.

[20] R. Greve, *On the response of the Greenland ice sheet to greenhouse climate change*, Climatic Change, 46 (2000), pp. 289–303.

[21] ———, *Glacial isostasy: Models for the response of the Earth to varying ice loads*, in Continuum Mechanics and Applications in Geophysics and the Environment, B. Straughan et al., eds., Springer, 2001, pp. 307–325.

[22] R. Greve, R. Takahama, and R. Calov, *Simulation of large-scale ice-sheet surges: The ISMIP-HEINO experiments*, Polar Meteorol. Glaciol., 20 (2006), pp. 1–15.

[23] P. Halfar, *On the dynamics of the ice sheets 2*, J. Geophys. Res., 88 (1983), pp. 6043–6051.

[24] R. C. A. Hindmarsh, *Thermoviscous stability of ice-sheet flows*, J. Fluid Mech., 502 (2004), pp. 17–40.

[25] ———, *Stress gradient damping of thermoviscous ice flow instabilities*, J. Geophys. Res., 111 (2006). `doi:10.1029/2005JB004019`.

[26] R. Hooke, *Flow law for polycrystalline ice in glaciers: comparison of theoretical predictions, laboratory data, and field measurements*, Rev. Geophys. Space. Phys., 19 (1981), pp. 664–672.

[27] C. L. Hulbe and D. R. MacAyeal, *A new numerical model of coupled inland ice sheet, ice stream, and ice shelf flow and its application to the West Antarctic Ice Sheet*, J. Geophys. Res., 104 (1999), pp. 25349–25366.

[28] A. Humbert, R. Greve, and K. Hutter, *Parameter sensitivity studies for the ice flow of the Ross Ice Shelf, Antarctica*, J. Geophys. Res., 110 (2005). doi:10.1029/2004JF000170.

[29] K. Hutter, *Theoretical Glaciology*, D. Reidel, 1983.

[30] P. Huybrechts and J. de Wolde, *The dynamic response of the Greenland and Antarctic ice sheets to multiple-century climatic warming*, J. Climate, 12 (1999), pp. 2169–2188.

[31] P. Huybrechts et al., *The EISMINT benchmarks for testing ice-sheet models*, Ann. Glaciol., 23 (1996), pp. 1–12.

[32] J. Imbrie and eight others, *The orbital theory of Pleistocene climate: Support from a revised chronology of the marine $\delta^{18}o$ record*, in Milankovitch and Climate, vol. 1, 1984, pp. 269–305.

[33] E. R. Ivins and T. S. James, *Antarctic glacial isostatic adjustment: a new assessment*, Antarctic Science, 17 (2005), pp. 537–549.

[34] D. Jenssen, *A three–dimensional polar ice–sheet model*, J. Glaciol., 18 (1977), pp. 373–389.

[35] J. Johnson and J. L. Fastook, *Northern Hemisphere glaciation and its sensitivity to basal melt water*, Quat. Int., 95 (2002), pp. 65–74.

[36] I. Joughin, M. Fahnestock, D. MacAyeal, J. L. Bamber, and P. Gogineni, *Observation and analysis of ice flow in the largest Greenland ice stream*, J. Geophys. Res., 106 (2001), pp. 34021–34034.

[37] I. Joughin, D. R. MacAyeal, and S. Tulaczyk, *Basal shear stress of the Ross ice streams from control method inversions*, J. Geophys. Res., 109 (2004). doi:10.1029/2003JB002960.

[38] A. Letrèguilly, P. Huybrechts, and N. Reeh, *Steady-state characteristics of the Greenland ice sheet under different climates*, J. Glaciol., 37 (1991), pp. 149–157.

[39] C. S. Lingle and J. A. Clark, *A numerical model of interactions between a marine ice sheet and the solid earth: Application to a West Antarctic ice stream*, J. Geophys. Res., 90 (1985), pp. 1100–1114.

[40] D. R. MacAyeal, *Large-scale ice flow over a viscous basal sediment: theory and application to ice stream B, Antarctica*, J. Geophys. Res., 94 (1989), pp. 4071–4087.

[41] D. R. MacAyeal, V. Rommelaere, P. Huybrechts, C. Hulbe, J. Determann, and C. Ritz, *An ice-shelf model test based on the Ross ice shelf*, Ann. Glaciol., 23 (1996), pp. 46–51.

[42] C. F. Maule, M. E. Purucker, N. Olsen, and K. Mosegaard, *Heat flux anomalies in Antarctica revealed by satellite magnetic data*, Science, 309 (2005), pp. 464–467.

[43] L. W. Morland, *Unconfined ice-shelf flow*, in Dynamics of the West Antarctic ice sheet, C. J. van der Veen and J. Oerlemans, eds., Kluwer Academic Publishers, 1987, pp. 99–116.

[44] L. W. Morland and R. Zainuddin, *Plane and radial ice-shelf flow with prescribed temperature profile*, in Dynamics of the West Antarctic ice sheet, C. J. van der Veen and J. Oerlemans, eds., Kluwer Academic Publishers, 1987, pp. 117–140.

[45] K. W. Morton and D. F. Mayers, *Numerical Solutions of Partial Differential Equations: An Introduction*, Cambridge University Press, second ed., 2005.

[46] A. Ohmura and N. Reeh, *New precipitation and accumulation maps for Greenland*, J. Glaciol., 37 (1991), pp. 140–148.

[47] W. S. B. Paterson, *The Physics of Glaciers*, Pergamon, 3rd ed., 1994.

[48] W. S. B. Paterson and W. F. Budd, *Flow parameters for ice sheet modeling*, Cold Reg. Sci. Technol., 6 (1982), pp. 175–177.

[49] A. Payne, *EISMINT: Ice sheet model intercomparison exercise phase two. Proposed simplified geometry experiments*. URL `http://homepages.vub.ac.be/~phuybrec/eismint/thermo-descr.pdf/`, 1997.

[50] A. Payne et al., *Results from the EISMINT model intercomparison: the effects of thermomechanical coupling*, J. Glaciol., 153 (2000), pp. 227–238.

[51] A. J. Payne and D. J. Baldwin, *Analysis of ice–flow instabilities identified in the EISMINT intercomparison exercise*, Ann. Glaciol., 30 (2000), pp. 204–210.

[52] W. R. Peltier, *The impulse response of a Maxwell earth*, Rev. Geophys. Space Phys., 12 (1974), pp. 649–669.

[53] W. R. Peltier, D. L. Goldsby, D. L. Kohlstedt, and L. Tarasov, *Ice–age ice–sheet rheology: constraints from the last Glacial Maximum form of the Laurentide ice sheet*, Ann. Glaciol., 30 (2000), pp. 163–176.

[54] M. Reed and B. Simon, *Methods of Modern Mathematical Physics I*, Academic Press, 2nd ed., 1980.

[55] C. Ritz, *EISMINT Intercomparison Experiment: Comparison of existing Greenland models*. `http://homepages.vub.ac.be/ phuybrec/eismint/greenland.html`, 1997.

[56] C. Ritz, A. Fabre, and A. Letréguilly, *Sensitivity of a Greenland ice sheet model to ice flow and ablation parameters: consequences for the evolution through the last glacial cycle*, Climate Dyn., 13 (1997), pp. 11–24.

[57] C. Ritz, V. Rommelaere, and C. Dumas, *Modeling the evolution of Antarctic ice sheet over the last 420,000 years: Implications for altitude changes in the Vostok region*, J. Geophys. Res., 102 (1997), pp. 12219–12233.

[58] P. Roache, *Verification and Validation in Computational Science and Engineering*, Hermosa Publishers, Albuquerque, New Mexico, 1998.

[59] V. Rommelaere and D. R. MacAyeal, *Large-scale rheology of the Ross Ice Shelf, Antarctica, computed by a control method*, Ann. Glaciol., 24 (1997), pp. 43–48.

[60] F. Saito, A. Abe-Ouchi, and H. Blatter, *European Ice Sheet Modelling Initiative (EISMINT) model intercomparison experiments with first-order mechanics*, J. Geophys. Res., 111 (2006). `doi:10.1029/2004JF000273`.

[61] ———, *An improved numerical scheme to compute horizontal gradients at the ice-sheet margin: its effect on the simulated ice thickness and temperature*, Ann. Glaciol., 46 (2007), pp. 87–96.

[62] C. Schoof, *A variational approach to ice stream flow*, J. Fluid Mech., 556 (2006), pp. 227–251.

[63] ———, *Variational methods for glacier flow over plastic till*, J. Fluid Mech., 555 (2006), pp. 299–320.

[64] N. M. Shapiro and M. H. Ritzwoller, *Inferring surface heat flux distributions guided by a global seismic model: particular application to Antarctica*, Earth and Planetary Science Letters, 223 (2004), pp. 213–224.

[65] L. Tarasov and W. R. Peltier, *Greenland glacial history and local geodynamic consequences*, Geophys. J. Int., 150 (2002), pp. 198–229.

[66] M. Weis, R. Greve, and K. Hutter, *Theory of shallow ice shelves*, Continuum Mech. Thermodyn., 11 (1999), pp. 15–50.

[67] P. Wesseling, *Principles of Computational Fluid Dynamics*, Springer-Verlag, 2001.

Appendix A. PISM command line options

Much of the behavior of PISM can be set at the command line by options. For example, the command

```
$  pismv -test C -Mx 61 -gk -e 1.2 -o foo
```

includes five options "`-test`", "`-Mx`", "`-gk`", "`-e`", and "`-o`", The first of these options includes a single character argument, the second an integer argument, the third has no argument, the fourth has a floating point argument, and the fifth has a string argument.

The format of the option documentation below is

"`-optionname` [*A*] [`B` *only*]: Description."

Here "A" is the default value and "`B`" is a list of the allowed executables. The option applies to all executables (`pismr`, `pisms`, `pismv`) unless the allowed executables are specifically stated by giving "[`B` *only*]".

As PISM is a PETSc program, all PETSc options are available [1]. See the next section, which recalls some of these PETSc options.

`-adapt_ratio` [*0.12*]: Adaptive time stepping ratio for the explicit scheme for the mass balance equation.

`-bed_def_iso`: Compute bed deformations by simple pointwise isostasy. Assumes that the bed at the starting time is in equilibrium with the load so the bed elevation is equal to the starting bed elevation minus a multiple of the increase in ice thickness from the starting time, roughly: $b(t,x,y) = b(0,x,y) - f[H(t,x,y) - H(0,x,y)]$. Here f is the density of ice divided by the density of the mantle. See Test H in Verification section.

`-bed_def_lc`: Compute bed deformations, caused by the changing load of the ice, using a viscoelastic earth model. Uses the model and computational technique described in [11], based on the continuum model in [39].

`-bif` [`pismr, pgrn` *only*]: The model can be "bootstrapped" from certain NetCDF files with just the right information. See section 9. Compare `-if`.

`-bif_legacy` [`pant` *only*]: The model can be "bootstrapped" from certain NetCDF files dealing with antarctica (`init.nc`).

`-constant_hardness`: If this option is used then the velocities in ice shelves and streams (see `-mv` below) are computed with a constant, temperature-independent hardness parameter $\bar{B}$. In particular, the viscous stress term in the x-component (for example) of the SSA equations for ice shelves and ice streams is

$$\frac{\partial}{\partial x}\left(2\nu H\left(2\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)\right)$$

where

$$\nu = \frac{\bar{B}}{2}\left[\left(\frac{\partial u}{\partial x}\right)^2+\left(\frac{\partial v}{\partial y}\right)^2+\frac{1}{4}\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)^2+\frac{\partial u}{\partial x}\frac{\partial v}{\partial y}\right]^{(1-n)/(2n)}.$$

Generally $\bar{B}$ is computed by a vertical-integral of a function of the temperature field. If `-constant_hardness` is used then $\bar{B}$ is set to the given value.

In the case of `pismd -ross`, which performs the EISMINT Ross ice shelf intercomparison, a constant value $\bar{B} = 1.9 \times 10^8\,\mathrm{Pa\,s}^{1/3}$ is the default [41].

`-constant_nu` [*30.0*]: If this option is used then the velocities in ice shelves and streams (see `-mv` below) are computed with a constant viscosity ν; see the option `-constant_hardness` above. Generally the viscosity is computed by a nonlinear iteration. With option `-constant_nu`, this iteration does not occur. If `-constant_nu` is set then `-constant_hardness` is ignored. The argument is given in units of MPa a, and the default value is 30 MPa a, the value given in [57].

`-ccl3` [`pgrn -forcing` *only*]: Run EISMINT-GREENLAND climate control experiment. See section 9

`-d`: Specifies diagnostic (X Windows) viewers. See Appendix C.

`-datprefix` [*PISM*] [`pisms -ismip H` *only*]: Specify base name for ISMIP-HEINO deliverable `.dat` files. See also `-no_deliver`.

`-dbig`: Specifies larger (about twice linear dimensions) diagnostic viewers. See Appendix C.

`-e` [*1.0*]: Flow enhancement factor.

`-eo` [`pismv` *only*]: Only evaluate the exact solution (don't do numerical approximation at all). See section 7.

`-eisII` [*A*] [`pisms` *only*]: Choose single character name of EISMINT II [50] simplified geometry experiment. Allowed values are A, B, C, D, E, F, G, H.

`-forcing` [`pgrn` *only*]: Uses a NetCDF file to apply changes in surface temperature and sea level. See section 9.

`-full3Dout`: Save the model state with additional full velocity field. That is, save all scalar components $u(x,y,z)$, $v(x,y,z)$, $w(x,y,z)$ of the velocity field for (x,y,z) everywhere in the three-dimensional computational box. Not needed when using `pismd`, for which saving the full three-dimensional velocity field is the default. Has the effect of (roughly) doubling the size of the output model state NetCDF file.

`-gk` [`pisms,pismr` *only*]: Sets the flow law to Goldsby-Kohlstedt. Same as `-law 4`. See `-law` for more complete option choice of flow law.

`-grad_from_eta`: The surface gradient is usually computed by simple centered-differences on the surface elevation. With this option it is computed by first transforming the thickness H by $\eta = H^{(2n+2)/n}$ and then differentiating the sum of the thickness and the bed:

$$\nabla h = \nabla H + \nabla b = \frac{n}{(2n+2)}\eta^{(-n-2)/(2n+2)}\nabla\eta + \nabla b.$$

Here b is the bed elevation and h is the surface elevation. Note that only $n = 3$ is used in this transformation regardless of the chosen flow law. The transformation may have benefits in that the surface value of the vertical velocity may be better behaved near the margin, e. .g. as in EISMINT II experiments, but this is not yet clear. See [14] for the technical explanation of this transformation and compare [61].

`-gwl3` [`pgrn` *only*]: Run EISMINT-GREENLAND greenhouse warming. See section 9

`-id`: Sets the x grid index at which a sounding diagnostic viewer (section C) is displayed. The integer argument should be in the range $0, \ldots,$ `Mx` $- 1$. The default is (`Mx` $- 1)/2$ at the center of the grid. For example, `pismv -test G -d t -id 10`.

`-if`: The model can be initialized (restarted) from a NetCDF file `foo.nc` written by the model, e.g. `foo.nc` from `-o foo -of n`. Compare `-bif`.

`-jd`: Sets the y grid index at which a sounding diagnostic viewer (section C) is displayed. The integer argument should be in the range $0, \ldots,$ `My` $- 1$. The default is (`My` $- 1)/2$ at the center of the grid. For example, `pismv -test G -d t -jd 10`.

`-kd` [*0*]: Sets the z grid index at which map-plane diagnositic views are shown. Compare `pismv -test G -Mz 101 -d T` and `pismv -test G -Mz 101 -d T -kd 50`. See section C.

`-law` [*0*]: Allows choice of thermocoupled flow law. The options are in table 14 below. Note that a "flow law" here means the function $F(\sigma, T, P, d)$ in the relation

$$\dot\epsilon_{ij} = F(\sigma, T, P, d)\, \sigma'_{ij}$$

where $\dot\epsilon_{ij}$ is the strain rate tensor, σ'_{ij} is the stress deviator tensor, T is the ice temperature, $\sigma^2 = \frac{1}{2}\|\sigma'_{ij}\|_F$ so σ is the second invariant of the stress deviator tensor, P is the pressure, and d is the grain size. That is, we are addressing isotropic flow laws only, and one can choose the scalar function. Note that the inverse form of such a flow law in needed for ice shelves and ice streams:

$$\sigma'_{ij} = 2\nu(\dot\epsilon, T, P, d)\, \dot\epsilon_{ij}$$

Here $\nu(\dot\epsilon, T, P, d)$ is the effective viscosity. The need for this inverse form of the flow law explains the "hybrid" law `-law 4` (or `-gk`).

`-Lx`: The x direction half-width of the computational box, in kilometers. See section 6.

`-Ly`: The y direction half-width of the computational box, in kilometers. See section 6.

`-Lz`: The height of the computational box for the ice (excluding the bedrock thermal model part). See section 6.

`-mato` [*pism_views*]: Specifies the name of the MATLAB-readable file in which to save the views. Use with `-matv`, to set which views to save; if `-matv` is not set then `-mato` has no effect.

`-matv`: Specifies which MATLAB "views" to save at end of run. See Appendix C for list of single character names of the views. The default is for *no* MATLAB views. See `-mato` for setting the name of the MATLAB-readable file in which to save the views. Typically the user will write `-mato foo -matv HT0c` to save four views in the MATLAB-readable ASCII file `foo.m`.

`-maxdt` [*60.0*]: The maximum time-step in years. The adaptive time-stepping scheme will make the time-step shorter than this as needed for stability, but not longer. See section 6.

`-Mbz` [*1*]: Number of grid points in the bedrock for the bedrock thermal model. The highest grid point corresponds to the base of the ice $z = 0$, and so `Mbz`> 1 is required to actually have bedrock thermal model. Note this option is unrelated to the bed deformation model (glacial isostasy model); see option `-bed_def` for that.

`-Mmax` [`pisms -eisII` *only*]: Set value of $M_{\max}$ for EISMINT II.

`-mu_sliding` [*3.17e-11*]: The sliding law parameter in SIA regions of the ice.

TABLE 14. Choosing the flow law.

Flow Law	**Option**	**Comments and Reference**
Paterson-Budd law	`-law 0`	Fixed Glen exponent $n = 3$. There is a split "Arrhenius" term $A(T) = A\exp(-Q/RT^*)$ where $(A = 3.615 \times 10^{-13}\,\mathrm{s}^{-1}\,\mathrm{Pa}^{-3}, Q = 6.0 \times 10^4\,\mathrm{J\,mol}^{-1})$ if $T^* < 263$ K and $(A = 1.733 \times 10^3\,\mathrm{s}^{-1}\,\mathrm{Pa}^{-3}$, $Q = 13.9 \times 10^4\,\mathrm{J\,mol}^{-1})$ if $T^* > 263$ K and where T^* is the homologous temperature [48].
Cold part of Paterson-Budd	`-law 1`	Regardless of temperature, the values for $T^* < 263$ K in the Paterson-Budd law above apply. This is the flow law used in the thermomechanically coupled exact solutions Tests **F** and **G** described in [10, 9] and run by `pismv -test F`, `pismv -test F`.
Warm part of Paterson-Budd	`-law 2`	Regardless of temperature, the values for $T^* > 263$ K in Paterson-Budd apply.
Hooke law	`-law 3`	Fixed Glen exponent $n = 3$. Here $A(T) = A\exp(-Q/(RT^*) + 3C(T_r - T^*)^\kappa)$; values of constants as in [26, 51].
Hybrid of Goldsby-Kohlstedt and Paterson-Budd	`-law 4`	Goldsby-Kohlstedt law with a combination of exponents from $n = 1.8$ to $n = 4$ [18] in grounded shallow ice approximation regions. Paterson-Budd flow for ice streams and ice sheets. See mask for SIA versus stream versus shelf by `-d m`.

`-Mx` [*61*]: Number of grid points in x horizontal direction.

`-My` [*61*]: Number of grid points in y horizontal direction.

`-Mz` [*31*]: Number of grid points in z (vertical) direction.

`-no_mass`: Forces PISM not to change the ice thickness. No time steps of the mass conservation equation are computed.

`-no_report` [`pismv` *only*]: Do not report errors at the end of a verification run.

`-no_spokes` [*0*]: The strain heating term can be smoothed by non-physical averaging the neighboring horizontal neighbors (those which are within the ice) [10]. The integer parameter controls the number of neighboring grid points over which the average is computed. For instance, `-no_spokes 0` is no smoothing while `-no_spokes 3` is smoothing over the 3-neighborhood of horizontal grid points, that is, over a distance of `3*dx`.

`-no_temp`: Do not change temperature or age values within the ice. That is, do not do time steps of energy conservation and age equation.

`-o` [*unnamed.nc*]: Give name of output file: `-o foo` writes an output file named `foo.nc`. See the description of option `-if`. Default name is `unnamed.nc` under `pismr`, `simp_exper.nc` under `pisms`, and `verify.nc` under `pismv`.

`-ocean_kill`: If used with input from a NetCDF initialization file which has ice-free ocean mask, will zero out ice thicknesses in areas that were ice-free ocean at time zero. Has no effect when used in conjunction with `-no_mass`.

`-of` [*n*]: Format of output file(s). Possible values are `n` for model state written to a NetCDF file `foo.nc` and `m` for selected variables written to an ASCII Matlab file `foo.m`. PISM can be restarted using `-if` from the output `.nc` files. Multiple files can be written, for instance, `-o foo -of nm` writes both `foo.nc` and `foo.m`.

`-pause` [`pismd -ross,pismv -test I,pismv -test J` *only*]: Pause for given number of seconds at the end of the run when the results are displayed.

`-pdd`: Turns on the positive degree day model (which is off by default for `pismr`, `pisms`, and `pismv`). See subsection 6.5.

`-pdd_factor_ice` [*0.008*]: The amount of ice, measured in meters, which is melted per positive degree day. In units of $\mathrm{m}\,^{\circ}\mathrm{C}^{-1}\mathrm{day}^{-1}$. See subsection 6.5.

`-pdd_factor_snow` [*0.003*]: The amount of snow, measured in meters ice-equivalent, which is melted per positive degree day. In units of $\mathrm{m}\,^{\circ}\mathrm{C}^{-1}\mathrm{day}^{-1}$. See subsection 6.5.

`-pdd_refreeze` [*0.6*]: Fraction of melted snow, from positive degree day melting, which locally refreezes as ice (and therefore adds to accumulation). See subsection 6.5.

`-pdd_repeatable`: Base the randomness specified by `pdd_std_dev` on a predictable seed. Makes no difference if `pdd_std_dev` is 0.0. Allows runs with positive `pdd_std_dev` to be identically repeated. See subsection 6.5.

`-pdd_std_dev` [*0.0*]: Standard deviation of temperature increment added each day to temperature cycle in positive degree day model. Units of °C. See subsection 6.5.

`-pdd_summer_warming` [*15.0*]: Difference between peak summer temperature and mean annual temperature at each grid location. Units of °C. See subsection 6.5.

`-plastic`: Use Schoof's plastic till model for ice streams at all grounded points on the ice sheet. Must be used along with `-ssa` to turn on the solution of the SSA (shallow shelf approximation). See subsection 4.3. See options `-plastic_reg`, `-till_cohesion`, `-till_friction_angle`, and `-till_pw_fraction`, all of which control the plastic till model in various ways.

`-plastic_reg` [*0.01 m/a*]: Set the value of ϵ regularization of plastic till; this is the second "ϵ" in formula (4.1) in [62]. Only effective if used with `-plastic` and `-ssa`.

`-regrid`: See section 6.

`-regrid_vars`: See section 6.

`-reg_length_schoof` [*1000 km*]: Set the "L" in formula (4.1) in [62]. To use the regularization described by Schoof, one must set `-mv_eps 0.0` to turn off the other regularization mechanism, otherwise there is a double regularization.

`-reg_vel_schoof` [*1 m/a*]: Set the *first* "ϵ" in formula (4.1) in [62]. To use the regularization described by Schoof, one must set `-mv_eps 0.0` to turn off the other regularization mechanism, otherwise there is a double regularization. Use `-plastic_reg` above to set the second "ϵ" in formula (4.1) of [62].

`-Rel` [`pisms -eisII` *only*]: Set value of R_{el} for EISMINT II.

`-ross` [`pisms` *only*]: Run the EISMINT Ross ice shelf validation [41]. Requires data from `http://homepages.vub.ac.be/~phuybrec/eismint/iceshelf.html`.

`-Sb` [`pisms -eisII` *only*]: Set value of S_b for EISMINT II.

`-ssa`: Use the equations of the shallow shelf approximation [40, 43, 62, 66] for ice shelves and dragging ice shelves (i.e. ice streams) where so-indicated by the mask. To view the mask use `-d m`. See also `-plastic` and `-super`.

`-ssa_eps` [*1.0e15*]: The numerical scheme for the shallow shelf approximation [66] computes an effective viscosity which which depends on velocity and temperature. After that computation, this constant is added to the effective viscosity (to keep it bounded away from zero). The units are kg $\text{m}^{-1}\,\text{s}^{-1}$.

In fact there is a double regularization by default because the Schoof regularization mechanism described in equation (4.1) of [62] is also used. Turn off this lower bound mechanise by `-ssa_eps 0.0` to exclusively use the Schoof regularization mechanism; see `-reg_vel_schoof` and `-reg_length_Schoof` below. Note `ssa_eps` is set to zero automatically when running `pismv -test I`.

`-ssa_rtol` [*1.0e-4*]: The numerical scheme for the shallow shelf approximation [66] does a nonlinear iteration wherein velocities (and temperatures) are used to compute a vertically-averaged effective viscosity which is used to solve the equations for horizontal velocity. Then the new velocities are used to recompute an effective viscosity, and so on. This option sets the relative change tolerance for the effective viscosity.

In particular, the nonlinear part of the iteration requires that successive values $\nu^{(k)}$ of the vertically-averaged effective viscosity satisfy

$$\frac{\|(\nu^{(k)} - \nu^{(k-1)})H\|_2}{\|\nu^{(k)}H\|_2} \le \text{ssa\_rtol}$$

in order to end the iteration with $\nu = \nu^{(k)}$. See also `-ksp_rtol`, a PETSc option below, as one may want to require a high relative tolerance for the linear iteration as well.

`-ssl2` [`pgrn` *only*]: Run EISMINT-GREENLAND steady state experiment level 2. See section 9.

`-ssl3` [`pgrn` *only*]: Run EISMINT-GREENLAND steady state experiment level 3. Use with option `-gk`: "`pgrn -ssl3 -gk`. See section 9.

`-ST` [`pisms -eisII` *only*]: Set value of S_T for EISMINT II.

`-super`: Superpose the velocity fields. That is, add the velocity fields which result from the SIA and SSA versions of the balance of momentum equations. Only effective if used with `-ssa`.

`-tempskip` [*1*]: Number of mass-balance steps to perform before a temperature step is executed. A maximum value of `-tempskip 5` is recommended to avoid too many CFL violations.

`-test` [*A*] [`pismv` *only*]: Choose which verification test to run by giving its single character name. See section 7.

`-till_cohesion` [*5.0e3*]: Cohesion used for computing the yield stress of the till in the plastic till model. Only effective if used with option `-plastic`. Units of Pa; note 5 kPa = 0.05 bar.

`-till_friction_angle` [*12*]: Friction angle used for computing the yield stress of the till in the plastic till model. Only effective if used with option `-plastic`. Units of degrees; note 12° gives $\mu = \tan(12°) = 0.21256$.

`-till_pw_fraction` [*0.95*]: Pore water pressure is this fraction of the overburden pressure when computing the yield stress of the till in the plastic till model. Only effective if used with option `-plastic`. Pure number (no units).

`-Tmin` [`pisms -eisII` *only*]: Set value of $T_{\min}$ for EISMINT II.

`-verbose`: Increased verbosity of standard output. Can be given without argument ("`-verbose`") or with a level which is one of the integers 0,1,2,3,4,5 ("`-verbose 2`"). The full scheme is given in table 15.

TABLE 15. Controlling the verbosity level to standard out.

Level	**Option**	**Meaning**
0	`-verbose 0`	never print to standard out *at all*; no warnings appear
1	`-verbose 1`	only warning messages and other high priority messages will appear
2	[`-verbose 2`]	default verbosity
3	`-verbose 3`	somewhat verbose; expanded description of grid at start
	or `-verbose`	and expanded information in summary
4	`-verbose 4`	
	or `-vverbose`	more verbose
5	`-verbose 5`	
	or `-vvverbose`	maximally verbose

`-vverbose`: See table 15.

`-vvverbose`: See table 15.

`-y` [*1000*]: Number of model years to run.

`-ye`: Model year at which to end the run.

`-ys` [*0*]: Model year at which to start the run.

Appendix B. PETSC command line options (for PISM users)

All PETSc programs accept command line options which control the manner in which PETSc distributes jobs among parallel processors, how it solves linear systems, what additional information it provides, and so on. The PETSc manual (`http://www.mcs.anl.gov/petsc/petsc-as/snapshots/petsc-current/docs/manual.pdf`) is the complete reference on these options. Here we list some that are perhaps most useful to PISM users.

`-da_processors_x`: Number of processors in x direction.

`-da_processors_y`: Number of processors in y direction.

`-display` [*A*]: s a *final* option, `-display :0` seems to frequently be needed to let PETSc use Xwindows when running multiple processes.

`-help`: Gives PISM help message and then a brief description of many PETSc options.

`-info`: Gives excessive information about PETSc operations during run. Option `-verbose` for PISM (above) is generally more useful, except possibly for debugging.

`-ksp_rtol` [*1e-5*]: For solving the ice stream and shelf equations with high resolution on multiple processors, it is recommended that this be tightened. For example,

```
$  mpiexec -n 8 pismv -test I -Mx 5 -My 769
```

works poorly on a certain machine, but

```
$  mpiexec -n 8 pismv -test I -Mx 5 -My 769 -ksp_rtol 1e-10
```

works fine.

`-ksp_type` [*gmres*]: Based on one processor evidence from `pismv -test I`, the following are possible choices in the sense that they work and allow convergence at some reasonable rate: `cg`, `bicg`, `gmres`, `bcgs`, `cgs`, `tfqmr`, `tcqmr`, and `cr`. It appears `bicg`, `gmres`, `bcgs`, and `tfqmr`, at least, are all among the best.

`-log_summary`: At the end of the run gives a performance summary and also a synopsis of the PETSc configuration in use.

`-pc_type` [*ilu*]: Several options are possible, but for solving the ice stream and shelf equations we recommend only `bjacobi`, `ilu`, and `asm`. Of these it is not currently clear which is fastest; they are all about the same for `pismv -test I` with high tolerances (e.g. `-ssa_rtol 1e-7 -ksp_rtol 1e-12`).

`-v`: Show version number of PETSc.

Appendix C. PISM viewers: Graphical and Matlab

Viewing the PISM state. Basic graphical views of the changing state of a PISM ice model are available at the command line by using the options "`-d`" and "`-dbig`" with additional arguments. For instance:

```
$  pismv -test G -d hTf -dbig 0
```

shows a map-plane views of surface elevation ("`h`"), temperature at the level specified by `-kd` ("`T`"), rate of change of thickness ("`f`") and of horizontal ice speed at the surface ("`0`", i.e. zero).

The option `-d` is followed by a space and then a list of single-character names of the diagnositic viewers. The option `-dbig` works exactly the same way, with the same list of single-character names available. The bigger viewers take precedence, so that "`-d hT -dbig T`" shows only two viewers, namely a regular size viewer for surface elevation and a larger viewer for temperature.

The same views of the PISM model can be saved *at the end of the run* to a Matlab-readable ASCII file by the same single character names. (At this point, *most* of the names are allowed; see the list below.) We will call these "Matlab views". For instance,

```
$  pismv -test G -mato foo -matv hTf0
```

will save surface elevation ("`h`"), temperature at the level specified by `-kd` ("`T`"), rate of change of thickness ("`f`") and of horizontal ice speed at the surface ("`0`") in an ASCII file `foo.m`. To use `foo.m` at the Matlab command line, make sure the Matlab path includes the directory with `foo.m`. Then just do

```
>> foo
```

You will be able to plot the surface elevation (for example) by

```
>> imagesc(x,y,flipud(H')), axis square, colorbar
```

but you can also use the Matlab tools `contour`, `surf`, and so on. (*The necessity to do* "`flipud(H')`" *to get the expected orientation, that is, the necessity of doing both do a transpose* and *a* `flipud`, *is for various deep reasons too complicated to explain here. Feel free to enquire, but* sorry *in the meantime.*)

Single character names (for each view). The single character diagnostic viewer and Matlab views names are:

`0`: Map-plane view of horizontal ice speed (magnitude of velocity) at the surface of the ice in meters per year.

`1`: Map-plane view of x-component of horizontal ice velocity at the *surface* of the ice in meters per year.

`2`: Map-plane view of y-component of horizontal ice velocity at the *surface* of the ice in meters per year.

`3`: Map-plane view of vertical ice velocity at the *surface* of the ice in meters per year; positive values are upward velocities.

`4`: Map-plane view of x-component of horizontal ice velocity at the *base* of the ice in meters per year.

`5`: Map-plane view of y-component of horizontal ice velocity at the *base* of the ice in meters per year.

`B`: Map-plane view of basal drag coefficient β for ice stream regions. β has units of Pa s m^{-1}. Displayed as log base ten of β.

`C`: Map-plane view of basal till yield stress τ_c, under plastic till ice stream model. Units of bar ($= 10^5$ Pa).

`D`: (*NOT available as a* MATLAB *view.*) Map-plane view of diffusivity coefficient D in mass balance equation in m^2/s. Meaningful only in regions of shallow ice flow.

`E`: Map-plane view of age of the ice, in years.

`F`: Map-plane view of basal geothermal heat flux, in milliWatts per meter squared.

`G`: Map-plane view of grain size, in millimeters. Displayed at chosen elevation above base; see option `-kd`.

`H`: Map-plane view of thickness in meters.

`L`: Map-plane view of basal melt water *thickness* in meters.

`P`: (*NOT available as a* MATLAB *view.*) (*ONLY available for* `pismv`.) Map-plane view of comPensatory heating term Σ_C in thermocoupled verification tests F and G. Displayed at chosen elevation above base; see option `-kd`.

`R`: Map-plane view of basal frictional heating in milliWatts per meter squared.

`S`: Map-plane view of strain heating term Σ in temperature equation, in Kelvin per year. Displayed at chosen elevation above base; see option `-kd`.

`T`: Map-plane view of absolute ice temperature in Kelvin. Displayed at chosen elevation above base; see option `-kd`.

`U`: Map-plane view of vertically averaged horizontal velocity in the x-direction *on the staggered grid* which is offset by positive one in i direction; in meters per year. (*Meaningful only in technical/numerical context*; use `u` generally.)

`V`: Map-plane view of vertically averaged horizontal velocity in the y-direction *on the staggered grid* which is offset by positive one in j direction; in meters per year. (*Meaningful only in technical/numerical context*; use `v` generally.)

`X`: Map plane view of x-component of horizontal velocity, in meters per year. Displayed at chosen elevation above base; see option `-kd`.

`Y`: Map plane view of y-component of horizontal velocity, in meters per year. Displayed at chosen elevation above base; see option `-kd`.

`Z`: Map plane view of vertical velocity, in meters per year. Displayed at chosen elevation above base; see option `-kd`.

`a`: Map-plane view of accumulation in meters per year.

`b`: Map-plane view of bed elevation in meters above sea level.

`c`: Map-plane view of horizontal speed, namely the absolute value of the vertically-averaged horizontal velocity. Displayed as log base ten of speed in meters per year.

`e`: Age in a vertical column (sounding); in years. See `-id`, `-jd` to set sounding location.

`f`: Map-plane view of thickening rate of the ice, in meters per year.

`g`: Grain size in a vertical column (sounding); in millimeters. See `-id`, `-jd` to set sounding location.

`h`: Map-plane view of ice surface elevation in meters above sea level.

`i`: Map-plane view of vertically-averaged effective viscosity times thickness; on i offset grid. Only meaningful in ice streams and shelves.

`j`: Map-plane view of vertically-averaged effective viscosity times thickness; on i offset grid. Only meaningful in ice streams and shelves.

`k`: (*NOT available as a* MATLAB *view.*) Iteration monitor for the Krylov subspace routines (KSP) in Petsc. Shows norm of residual versus iteration number. Has same effect as PETSc option `-ksp_monitor_draw`.

`l`: Map-plane view of basal melt *rate* in meters per year.

`m`: Map-plane view of mask for flow type: **1** = grounded shallow ice sheet flow, **2** = dragging ice shelf, **3** = floating ice shelf, 7 = ice free ocean (in original input file).

`n`: Map-plane view of the log base ten of the vertically-averaged effective viscosity times thickness on the regular grid. Only meaningful in ice streams and shelves.

`p`: Map-plane view of bed uplift rate in meters per year.

`q`: Map-plane view of basal sliding speed. Displayed as log base ten of speed in meters per year.

`r`: Map-plane view of surface temperature in Kelvin.

`s`: Strain heating term Σ in vertical column (sounding). See `-id`, `-jd` to set sounding location.

`t`: Absolute ice temperature in vertical column (sounding). Note this sounding extends into the bedrock, unlike the other soundings (e.g. `-d egsxyz`). See `-id`, `-jd` to set sounding location.

`u`: Map-plane view of vertically averaged horizontal velocity in the x-direction; in meters per year.

`v`: Map-plane view of vertically averaged horizontal velocity in the y-direction; in meters per year.

`x`: x-component of horizontal velocity in vertical column (sounding). See `-id`, `-jd` to set sounding location.

`y`: y-component of horizontal velocity in vertical column (sounding). See `-id`, `-jd` to set sounding location.

`z`: Vertical velocity (w-component of velocity) in vertical column (sounding). See `-id`, `-jd` to set sounding location.